Journal of open-source software.

https://joss.theoj.org/

Requirements:

https://joss.readthedocs.io/en/latest/submitting.html#submission-requirements

* ~~The software should be open source as per the~~[~~OSI definition~~](https://opensource.org/osd)~~.~~
* ~~The software should have an~~**~~obvious~~**~~research application.~~
* ~~You should be a major contributor to the software you are submitting.~~
* ~~Your paper should not focus on new research results accomplished with the software.~~
* Your paper (paper.md and BibTeX files, plus any figures) must be hosted in a Git-based repository together with your ~~software (although they may be in a short-lived branch which is never merged with the default).~~

In addition, the software associated with your submission must:

* ~~Be stored in a repository that can be cloned without registration.~~
* ~~Be stored in a repository that is browsable online without registration.~~
* ~~Have an issue tracker that is readable without registration.~~
* ~~Permit individuals to create issues/file tickets against your repository.~~
* A list of the authors of the software and their affiliations, using the correct format (see the example below).
* ~~A summary describing the high-level functionality and purpose of the software for a diverse, non-specialist audience.~~
* A Statement of Need section that clearly illustrates the research purpose of the software.
* A list of key references, including to other software addressing related needs. Note that the references should include full names of venues, e.g., journals and conferences, not abbreviations only understood in the context of a specific discipline.
* ~~Mention (if applicable) a representative set of past or ongoing research projects using the software and recent scholarly publications enabled by it.~~
* Acknowledgement of any financial support.

Max 1000 Words.

**Introduction**.

The digitisation of features in remotely sensed data to create mapped products or for training data in a supervised image classification is a time consuming, labour intensive and error prone task.

For this reason, the manual digitisation of features in operational projects is a limiting factor due to the time required to complete the task.

Consequently, there is a need for a simple to use region growing digitising tool which can be used to speed this process up and remove any human error.

Other region growing tools exist in commercial GIS software, such as in ERDAS Imagine (ERDAS 2015) and ArcGIS (ESRI 2018); as well as the region growing tool within the Semi-Automatic Classification Plugin (Congedo 2019) available for QGIS.

The solutions within the commercial software require a license which can be prohibitively expensive for some applications.

The freely available Semi-Automatic Classification QGIS plugin (Congedo 2019) requires dependencies external to QGIS and can be very difficult for non-experts to use particularly when used with UAV imagery or modified remotely sensed data.

The region growing plugin presented here was designed at the outset to be user friendly, computationally efficient, completely open source and can be used with any multiband remotely sensed imagery.

The plugin also requires no dependencies that are external to the original installation of QGIS, and can be used on Linux, MacOS and Windows.

# **Statement of Need**

# **Ongoing Projects Using this Software**

Zanzibar SIS

# **Methodology.**

Step 1.

Remotely sensed imagery is loaded into QGIS, and a check performed on the coordinate system, and changed to a UTM, metres-based coordinate system.

If the input imagery is acquired from a drone then it is converted from an RGB colour space into an *LAB* colour space, where colour is represented by a luminosity channel (*l*), and the colour on an axis of red – green (*a*) and blue-yellow (*b*) (Rathore, Kumar, and Verma 2012; Baldevbhai 2012; Pandey 2017). The LAB colour space also has the advantage of device display independence, and can correct for any uneven distribution of RGB values within the colour image (Niu et al. 2014). The RGB drone image is presented to the user in QGIS, and the l\*a\*b\* colour space image is used for processing.

When the plugin is using optical multispectral data or SAR imagery no colour transformation takes place, the measured pixel values are used.

A picture containing text

Description automatically generated

Figure 1. RGB Drone imagery converted into an l\*a\*b\* colour space.

Step 2.

When the user clicks within the remotely sensed imagery the coordinates of the clicked point are retrieved and a square neighbourhood created using the user defined distance between the clicked point and each edge of the square.

A picture containing text, tree, envelope

Description automatically generated

Figure 2. Location of user click within drone imagery and associated neighbourhood of pixels.

Step 3.

All pixels within the neighbourhood around the clicked point are plotted on a 3D feature space, and the Euclidean distance between each pixel within the feature space to the colour of the clicked pixel is then found.

Chart, scatter chart

Description automatically generated

Text

Description automatically generated with low confidence

Pixels with a lower colour distance are of a similar colour and could be considered as the same object.

A spatial weighting is also calculated where pixels which are spatially more distant from the clicked pixel are weighted higher. This reduces the change that pixel of the same colour but are a different object are selected.

Step 4.

The colour distance and spatial distance are summed and pixels with a combined value greater than the user defined threshold are masked out, leaving only candidate pixels which could be the same object.

Step 5.

Candidate pixels are vectorised forming features which cover similar pixels where the user has clicked. It is likely there will be outlying or disconnected pixels which are of similar colour but are not the same object as clicked by the user.

An intersection is performed between the vectorised pixels and the point where the user clicked, leaving only a feature which intersects the point clicked by the user.

Step 6.

The geometry of this feature is then simplified, broken geometries fixed and holes within the feature are filled. If desired the feature is buffered by a user defined amount and the final feature is committed to an output vector dataset as the digitised feature.

Graphical user interface, application

Description automatically generated

# **Funding**

This Software Development was funded as part of the Zanzibar SIS project and as part of a PhD Research Project supported by Aberystwyth University

# **References**

Baldevbhai, Patel Janakkumar. 2012. “Color Image Segmentation for Medical Images Using L\*a\*b\* Color Space.” *IOSR Journal of Electronics and Communication Engineering*. https://doi.org/10.9790/2834-0122445.

Congedo, Luca. 2019. “Semi-Automatic Classification Plugin Documentation Release 6.2.0.1.” *Release*.

ERDAS. 2015. “ERDAS Imagine Suite 2015.” Norcross: Hexagon Geospatial.

ESRI. 2018. “ArcGIS Desktop.” Redcross: Environmental Systems Research Institute.

Niu, Xiaojing, Meili Wang, Xianqiang Chen, Shihui Guo, Hongming Zhang, and Dongjian He. 2014. “Image Segmentation Algorithm for Disease Detection of Wheat Leaves.” In *International Conference on Advanced Mechatronic Systems, ICAMechS*. https://doi.org/10.1109/ICAMechS.2014.6911663.

Pandey, Manoj. 2017. “RGB to CIELab Color Space Conversion.” GitHub. https://gist.github.com/manojpandey/f5ece715132c572c80421febebaf66ae.

Rathore, Vivek Singh, Messala Sudhir Kumar, and Ashwini Verma. 2012. “Colour Based Image Segmentation Using L \* A \* B \* Colour Space Based On Genetic Algorithm.” *International Journal of Emerging Technology and Advanced Engineering*.