




1 The Region Growing Plugin (RegionGrow)

2 **Gregory Oakes^{*1}, Andy Hardy¹, and Yussuf Said Yussuf^{2, 3}**

3 **1** Department of Geography and Earth Sciences, Aberystwyth University, Aberystwyth SY23 3DB,
4 **UK 2** The State University of Zanzibar (SUZA), Tunguu, P.O.Box 146, Zanzibar **3** Tanzania Flying
5 Labs

DOI: [10.21105/joss.03279](https://doi.org/10.21105/joss.03279)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Pending Editor](#) 

Submitted: 11 May 2021

Published: 13 May 2021

License

Authors of papers retain
copyright and release the work
under a Creative Commons
Attribution 4.0 International
License ([CC BY 4.0](#)).

6 Summary

7 In the fields of image analysis, computer vision and remote sensing there are a number of
8 instances where the digitisation of features by the human operator is needed to develop mapped
9 products or for generating training data for classification routines. This process is notoriously
10 time consuming, labour intensive and error prone and, as such, the manual digitisation of
11 features in operational projects is often unfeasible, particularly in time-sensitive applications
12 like near real-time monitoring applications, emergency response mapping or where applications
13 are made at a continental or global scale. Consequently, there is a need for a simple to use
14 region growing tool that can be used to speed up the digitisation process and minimise human
15 error.

16 We present a new Region Growing plugin tool (RegionGrow) for the open source Geographical
17 Information System software QGIS. RegionGrow is designed to be user-friendly, computa-
18 tionally efficient, freely available, with functionality to accommodate a variety of sources of
19 multiband remotely sensed imagery. By selecting a pixel in the centre of an object of interest,
20 e.g. a forest, water body etc, RegionGrow generates connects pixels with a similar colour based
21 on its Euclidean distance with a L*A*B transformed colour space. This rapidly speeds up the
22 manual digitising process and minimises error.

*Corresponding Author

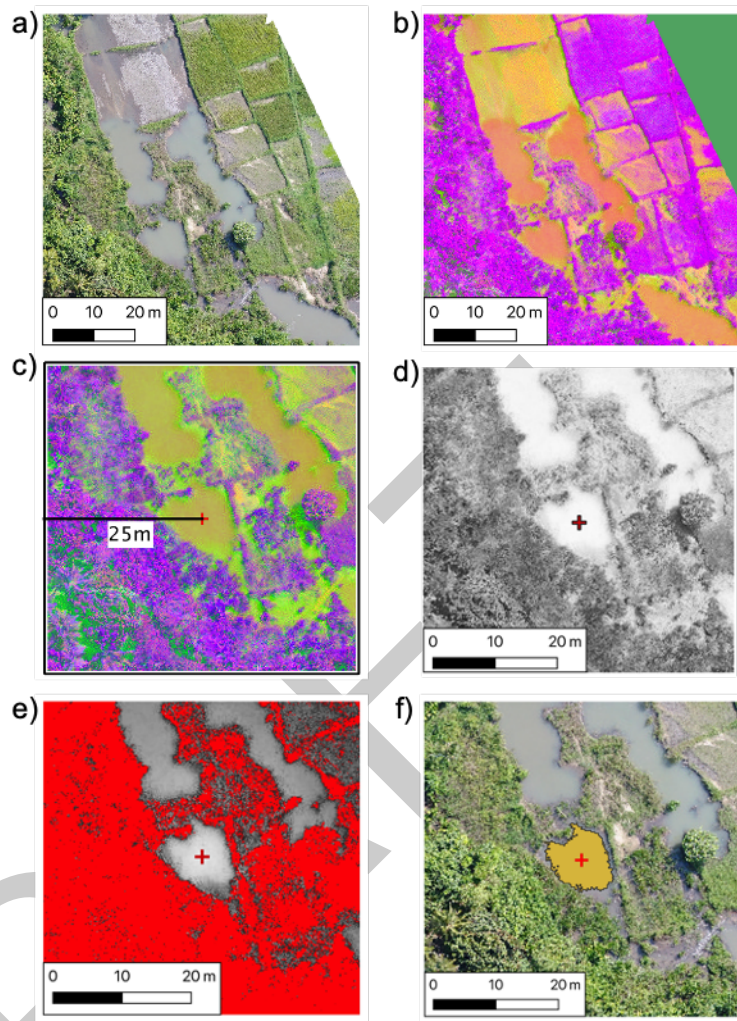


Figure 1: Overview of the processing steps performed by RegionGrow

Statement of Need

Within the open source software domain, there is just one region growing tool available from within the Semi-Automatic Classification (SAC) QGIS plugin (Congedo, 2016). However, this is intended to be used within the SAC classification workflow and cannot be used intuitively for other digitising needs.

An independent region growing tool is needed for two primary reasons: 1) Where users wish to collect training data for conducting image classifications outside of the SAC plugin tool, for instance, taking advantage of the most current and robust machine learning solutions from within Python libraries like TPOT (Olson & Moore, 2016) or Scikit-Learn (Pedregosa et al., 2011). 2) Where users wish to extract thematic information directly rather than following a classification routine.

Ongoing Projects

The RegionGrow QGIS plugin tool was originally developed for use within the project: Spatial Intelligence System (SIS) for precision larviciding based in Zanzibar, United Republic of Tanzania. Here, UAV technology is being used to map pools of water where malarial mosquitoes breed. By mapping these targets, they can be located by ground teams and treated with low-toxicity larvicide in an effort to reduce the malarial mosquito population. Although high resolution images (<10 cm) of areas of interest can be produced rapidly using UAVs, identifying and digitising water body features is a laborious and error-strewn process. Using RegionGrow, non-GIS specialists can now rapidly and accurately digitise these features, supplying vital information to ground-teams significantly quicker than following a conventional manual digitising approach (on average four times quicker).

Using this Software

Step 1.

Remotely sensed imagery (e.g. UAV/satellite imagery) is loaded into QGIS, ensuring that it is projected into a coordinate system (with map units in metres), e.g. Universal Transverse Mercator (UTM).

Input RGB imagery (typically acquired using commercially available UAV platforms) is converted into an L^*A^*B colour space, where colour is represented by a luminosity channel (L), the colour on a red – green axis (a) and a blue-yellow axis (b) (Baldevbhai & Anand, 2012; Pandey, 2017; Rathore et al., 2012). The LAB colour space can help account for the uneven distribution of RGB values within the colour image (Niu et al., 2014). Other imagery, i.e. multispectral optical imagery or radar backscatter imagery, do not undergo colour transformation.

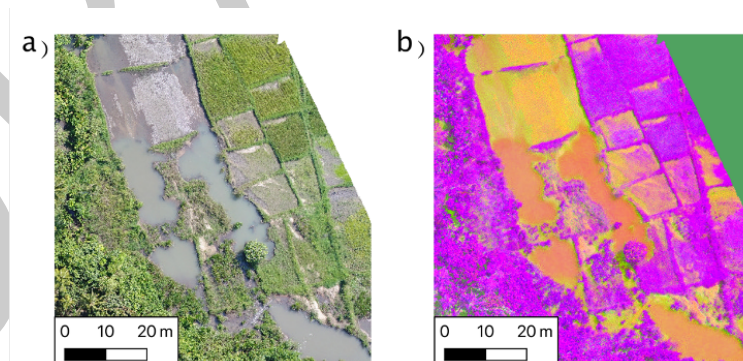


Figure 2: a) Example RGB Drone imagery. b) The same drone imagery converted into an $L^*A^*B^*$ colour space.

An output vector filename is defined by the user before continuing to the next step.

Step 2.

The user selects/clicks on a point of interest within the image. The coordinates of the selected point is retrieved and a square neighbourhood created (based on a user defined distance).

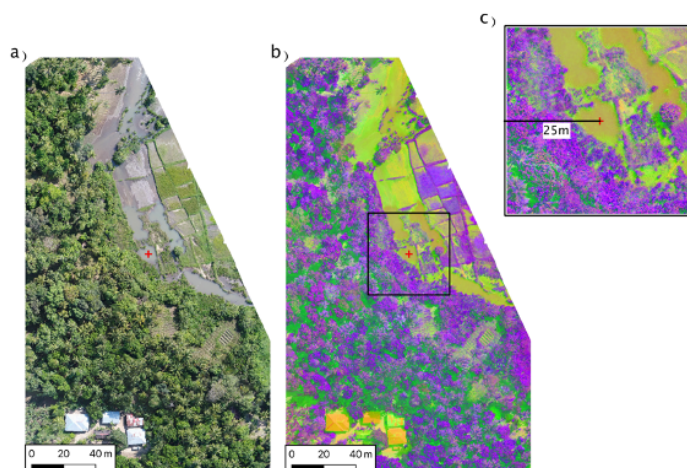


Figure 3: a). Location of user click within RGB drone imagery. B) L*A*B* transformed drone imagery and the location of the user clicked location. C) Local neighbourhood around in this case 25m around the user clicked location within the drone imagery.

Step 3.

All pixels within the neighbourhood are plotted into a 3D feature space. The Euclidean colour distance between each pixel and the selected pixel is determined. Pixels with a lower colour distance are considered to be of a similar colour.

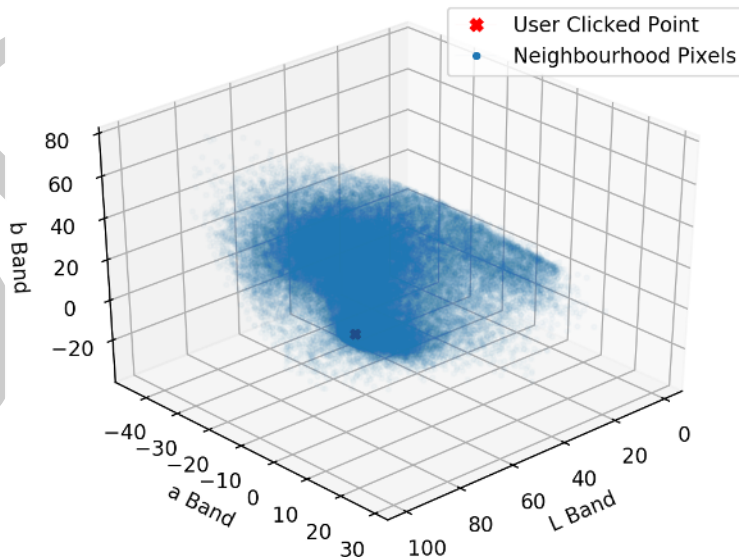


Figure 4: 3D feature space showing pixels within the local area of the user clicked location. The cross within the plot marks the user clicked pixel.

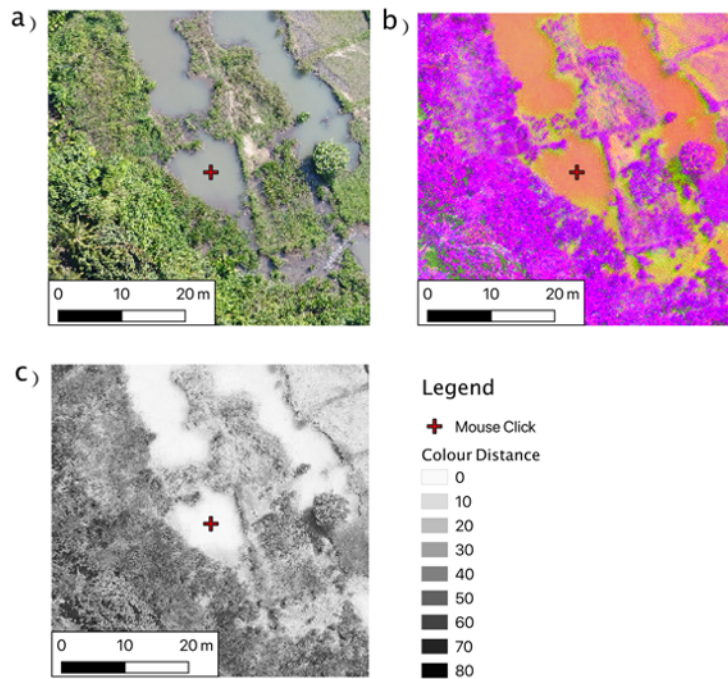


Figure 5: a) RGB Drone Imagery, b) L*A*B* transformed drone imagery and the location of the user clicked location. C) Euclidean distance within the 3D feature space between the user selected pixels and all pixels within the local neighbourhood of pixels.

A spatial weighting is also applied where pixels geographically further from the selected pixel have a lower weighting.

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 pixels considered to have a similar enough colour to be joined into the same region.

Step 5.

The output region (pixels of similar colour) are vectorised where they intersect the originally selected pixel, i.e. they must be joined to the selected pixel.

Step 6.

The geometry of this feature is then simplified, broken geometries fixed and holes within the feature are filled, and a buffer applied if defined by the user. The final region is committed to an output vector dataset (e.g. ESRI Shapefile, GeoJSON) as the digitised feature.

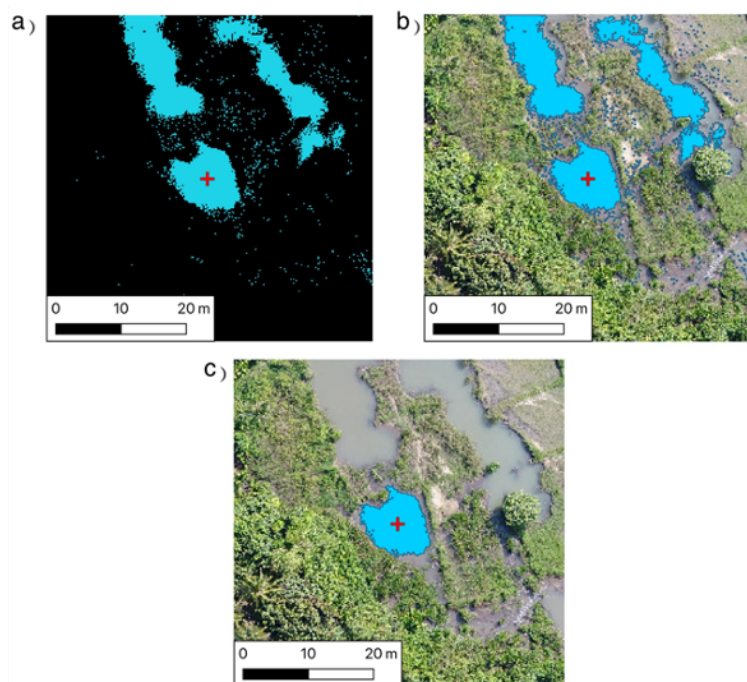


Figure 6: a) Pixels are selected where the colour distance is less than a user defined threshold. 6b) Selected pixels are vectorised and the feature which directly intersects the user clicked location is selected and simplified (c).

Dependencies

There are no dependencies required for the RegionGrow QGIS plugin tool beyond the dependencies required and installed by QGIS.

Funding

This Software Development was funded by the Bill and Melinda Gates Foundation Innovation Fund INV-010583 through a grant managed by the Innovative Vector Control Consortium (IVCC) and the Aberystwyth University Doctoral training programme.

References

- Baldevbhai, P. J., & Anand, R. (2012). Color image segmentation for medical images using $L^* a^* b^*$ color space. *IOSR Journal of Electronics and Communication Engineering*, 1(2), 24–45.
- Congedo, L. (2016). Semi-automatic classification plugin documentation. *Release*, 4(0.1), 29.
- Niu, X., Wang, M., Chen, X., Guo, S., Zhang, H., & He, D. (2014). Image segmentation algorithm for disease detection of wheat leaves. *Proceedings of the 2014 International Conference on Advanced Mechatronic Systems*, 270–273.

- 93 Olson, R. S., & Moore, J. H. (2016). TPOT: A tree-based pipeline optimization tool for
94 automating machine learning. *Workshop on Automatic Machine Learning*, 66–74.
- 95 Pandey, M. (2017). *RGB to CIE Lab color space conversion*. IEEE. [https://gist.github.com/
96 manojpandey/f5ece715132c572c80421febeba66ae](https://gist.github.com/manojpandey/f5ece715132c572c80421febeba66ae)
- 97 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel,
98 M., Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine
99 learning in python. *The Journal of Machine Learning Research*, 12, 2825–2830.
- 100 Rathore, V. S., Kumar, M. S., & Verma, A. (2012). Colour based image segmentation using
101 $l^* a^* b^*$ colour space based on genetic algorithm. *International Journal of Emerging
102 Technology and Advanced Engineering*, 2(6), 156–162.

DRAFT