remote sensing

MDPI

Article

Reproducibility and Practical Adoption of GEOBIA with Open-Source Software in Docker Containers

Christian Knoth * and Daniel Nüst

Institute for Geoinformatics, University of Münster, Heisenbergstraße 2, 48149 Münster, Germany; daniel.nuest@uni-muenster.de

* Correspondence: christian.knoth@uni-muenster.de; Tel.: +49-251-833-3056

Academic Editors: Norman Kerle, Markus Gerke, Sébastien Lefèvre and Prasad S. Thenkabail
Received: 30 December 2016; Accepted: 6 March 2017; Published: 18 March 2017

Abstract: Geographic Object-Based Image Analysis (GEOBIA) mostly uses proprietary software, but the interest in Free and Open-Source Software (FOSS) for GEOBIA is growing. This interest stems not only from cost savings, but also from benefits concerning reproducibility and collaboration. Technical challenges hamper practical reproducibility, especially when multiple software packages are required to conduct an analysis. In this study, we use containerization to package a GEOBIA workflow in a well-defined FOSS environment. We explore the approach using two software stacks to perform an exemplary analysis detecting destruction of buildings in bi-temporal images of a conflict area. The analysis combines feature extraction techniques with segmentation and object-based analysis to detect changes using automatically-defined local reference values and to distinguish disappeared buildings from non-target structures. The resulting workflow is published as FOSS comprising both the model and data in a ready to use Docker image and a user interface for interaction with the containerized workflow. The presented solution advances GEOBIA in the following aspects: higher transparency of methodology; easier reuse and adaption of workflows; better transferability between operating systems; complete description of the software environment; and easy application of workflows by image analysis experts and non-experts. As a result, it promotes not only the reproducibility of GEOBIA, but also its practical adoption.

Keywords: reproducibility; GEOBIA; Docker; conflict monitoring; reproducible research; object-based image analysis; QGIS; containerization

1. Introduction

1.1. Motivation

For a scientific method to gain impact in research, it must be understandable and replicable by fellow scientists. To gain impact in practice, it also needs to be easy to adopt by users from different fields. In computational sciences, such as Geographic Object-Based Image Analysis (GEOBIA), replicability requires access to code and data. Practical applicability in addition requires ease of use and the customizability of methods. In this work, we present a novel solution for the GEOBIA community to conduct research in a reproducible way by packaging code, data and even the required runtime environment in executable units called containers. We discuss the dilemma of balancing reproducibility, ease of use and the customizability of containerized methods and propose an approach to enable interaction with those methods through parameterized containers and a graphical user interface. The solutions can foster reproducibility and practical adoption of complex workflows, not least because they rely on Free and Open-Source Software (FOSS).

1.2. Reproducible Research

Openness in research is not a new topic, but there is a recent trend towards transparency and availability under the terms Open Science and Open Access (cf. [1]). All stakeholders in the research process contribute rules, incentives or guidelines to foster openness: on the funding side, the European Union requires Open Access as part of the Horizon 2020 framework programme (cf. [2]) and builds the European Open Science Cloud [3]; on the publishing side, journals such as Science [4] and Bioinformatics [5] encourage reproducibility; on the research institution side, scientists themselves argue for reproducibility with "Five selfish reasons to work reproducibly" [6], develop guidelines like the Vienna principles [7], or publish "Ten Simple Rules for Reproducible Computational Research" arguing in favour of a proper scientific workflow simply to be able to reproduce *your own* results [8]. A core notion of all of these activities is the ideal to publish data, methods, and software along with scholarly publications.

A definition of the term reproducibility is far from trivial, not least because it is often used together with other terms to describe different levels of redoing. The Vienna Principles' definition focuses on traceability [7]; others treat "reproducibility" and "replicability" as interchangeable [9] or completely different terms [10]; others qualify the term further, e.g., "computational reproducibility" [5].

For the remainder of this work, we use "reproduce", "reproducibility" or "computational reproducibility" to say a third party can re-run an analysis using code and data provided by the author of a published work, and this execution creates the same computational result (following the definition by Peng [5]). Besides technical difficulties, which is the focus of this work, computational reproducibility is likewise a question of proper practices. For example, researchers must fix seeds of random number generators [8], design transparent processing workflows [11] or utilize virtualization [12]. Nevertheless, the uniqueness of data (e.g., it can only be captured once by one sensor) or processing environments (e.g., supercomputers) can make "real" replication of results impossible, so that trust in the applied methods must be established instead [13]. In (GE)OBIA, the term "reproducible" is thus far used to describe a shift from manual analysis (based on interaction with a user interface) to script-based analysis using sets of processing steps and classification rules (cf. [14–16]).

Packaging data, code and documentation for reproducibility was previously described under the term research compendia by means of programming language-specific packaging mechanisms [9,17] or as research objects with a focus on workflows using semantic enrichment and provenance [18]. Alternative approaches to create self-contained packages for reproducibility are tracing techniques (cf. [19,20]).

1.3. FOSS for GEOBIA

To achieve trust in computational reproducibility, open-sourcing of workflows and the underlying software is crucial. While benefits of Free and Open-Source Software for business and security have been documented widely (see for example [21] and [22]), the important aspect of FOSS in science is the potential for scrutiny and collaboration. FOSS facilitates audits down to the level of the source code. It also promotes the reuse, improvement and adaption of a methodology or software. FOSS licensing models (for a quick introduction we recommend http://choosealicense.com/) allow to combine individual contributions of functional parts into a solution for a larger problem at hand. FOSS projects must allow (technically and legally) maintenance and re-purposing by third parties. This modularity lies at the roots of many open-source software projects and is propagated by the *Unix philosophy* [23]: Each programme should only provide a specific feature and excel at it.

A number of publications at GEOBIA conferences over the last few years demonstrate the growing interest in an FOSS approach to GEOBIA. For example, [24] developed a workflow for urban Land Use and Land Cover (LULC) classification using GRASS GIS and R. They also published the analysis reproducibly as an executable notebook. The work in [25] used the Orfeo ToolBox and R for automated selection of segmentation parameters and classification of urban scenes. The work in [26] applied

GRASS GIS to map impervious surfaces using aerial images and LIDAR (Light Detection and Ranging) data. Specific OBIA FOSS projects exist, as well; for example GeoDMA [27] and InterIMAGE [28] for desktop environments or InterCloud [29] for cloud computing infrastructures. Nevertheless these activities are still in the minority.

1.4. Balancing Reproducibility and Customizability for Practical Adoption

Complex workflows based on FOSS often integrate a number of independent and interdependent tools. These building blocks are developed by different communities and in differing maintenance cycles. Their use can be visible or invisible (e.g., transitive dependencies) to the analyst. This hampers reproducibility because of potential compatibility conflicts between different software packages in changing versions. Containerizing a computational method with all required software and data mitigates these problems because it eliminates the need to recreate the original runtime environment on the executing computer. In addition to the positive implications for research (see Section 1.2), we see a high potential in this approach to foster practical adoption of new GEOBIA methods. One main barrier to the application of GEOBIA approaches by practitioners besides cost factors are technical challenges in adopting complex analysis methods for their use case. Containerization can significantly simplify this process by providing non-experts with easy access to complex methods.

However, another important aspect with regard to practical adoption is the transferability of a method, i.e., the applicability to different study areas or different image types. GEOBIA facilitates the development of adaptive and transferable workflows, e.g., through the inclusion of context knowledge and human semantics [30]. There has been significant progress in the GEOBIA community concerning transferability, e.g., through automatic determination of segmentation parameters [31,32] or classification thresholds [33] and fuzzy classification [34]. However, although the workflows are increasingly robust, some adaptions (e.g., of segmentation parameters or classification thresholds) are usually necessary to tune an analysis method to a specific study area or image type [35,36]. Therefore, the possibility to customize a GEOBIA method is an important requirement for practical adoption, which can be hampered if analysis tools are containerized as black boxes without manipulation options. To reconcile reproducibility and customizability and thereby facilitate practical adoption, containerization should allow one to change model parameters or to use one's own input data in a containerized method. For widespread adoption, a Graphical User Interface (GUI) is required.

1.5. Contribution and Overview

The main contribution of this work is a fully-reproducible and open workflow for Geographic Object-Based Image Analysis (GEOBIA). We package a GEOBIA example study including the data and specific combination of software in an executable container. The container is parameterized so that the analysis can be evaluated and adapted. Users can interact with the container and manipulate an analysis through a GUI. The container is based on mainstream IT technology, and the software is a collection of general-purpose scripts, image analysis libraries and Geographic Information Systems (GIS). All components are FOSS. The implemented method detects destruction in a bitemporal image subset of a conflict area and builds on previous work [37,38]. The application of remote sensing for monitoring of human rights issues has been explored in a number of studies, and GEOBIA has proven to be a promising tool, e.g., for refugee camp monitoring [39,40] or damage assessment [41]. A comprehensive review of research and applications in this field (pixel- as well as object-based analyses) can be found in [42]. An open and transparent approach is specifically advantageous in human rights fact-finding because non-profit organizations face budget restrictions and because transparency is crucial when using complex techniques in a politically sensitive environment [43].

The following section describes the image analysis workflow, how it is implemented with FOSS and how it is made reproducible (Section 2). Section 3 explains the execution of the packaged analysis. Finally, we discuss the solution and the challenges (Section 4), and conclude with a summary and

outlook in Section 5. This paper is an extended version of our contribution to the GEOBIA 2016 conference [44].

2. Materials and Methods

2.1. Data

Two images of a village in Darfur, Sudan, are our example data. They are available online as part of a blog post by the American Association for the Advancement of Science (AAAS) Geospatial Technologies Project [45]. The copyright holder DigitalGlobe granted permission to re-publish them as part of this work. The images are previews of remote sensing imagery of the village Jonjona, located at 13.686°N, 24.979°E, before (December 2004) and after (February 2007) reported attacks in the area (see Figure 1). They were downloaded from the website in *jpg* format, manually georeferenced (entering coordinates retrieved from georeferenced imagery), resampled to a spatial resolution of 0.5 metres (nearest neighbour resampling), and saved as GeoTIFF files. The spatial resolution approximates the one of current commercial very high resolution satellites.

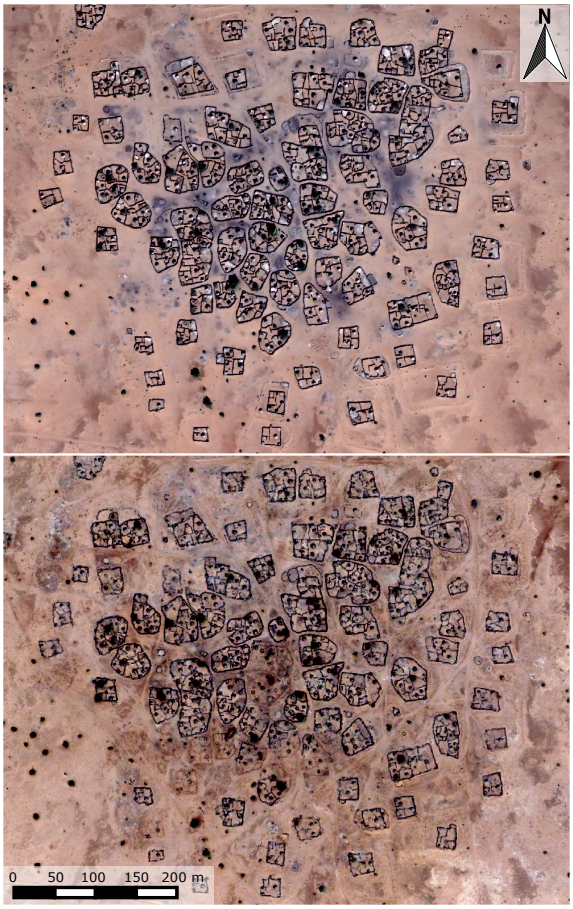

Figure 1. Example image of a village in Darfur (location: 13.686°N, 24.979°E) before (December 2004, top) and after (February 2007, bottom) a reported attack. Images © 2016 DigitalGlobe.

2.2. Example Analysis: Conflict Damage Assessment

2.2.1. Summary

The analysis model applies a rule-based approach to investigate bi-temporal images at two different scales represented by different image object levels. It demonstrates image interpretation using object-based features (e.g., shape, topological and hierarchical relations) using FOSS and is a

simplified version of the method described in [38]. In the first stage, the imagery is automatically searched for areas where settlements exist in the pre-conflict image (see Section 2.2.2). In the second stage, the change analysis is conducted on a finer segmentation level (see Section 2.2.3) within detected settlement areas. In these areas, the model identifies changed dwelling structures using relative change values, as well as shape and size. As demonstrated in [38], this approach facilitates the application in complex analyses of images with varying properties (such as sensor configurations, illumination conditions), especially when spectral features are investigated. Figure 2 shows an overview of the analysis workflow.

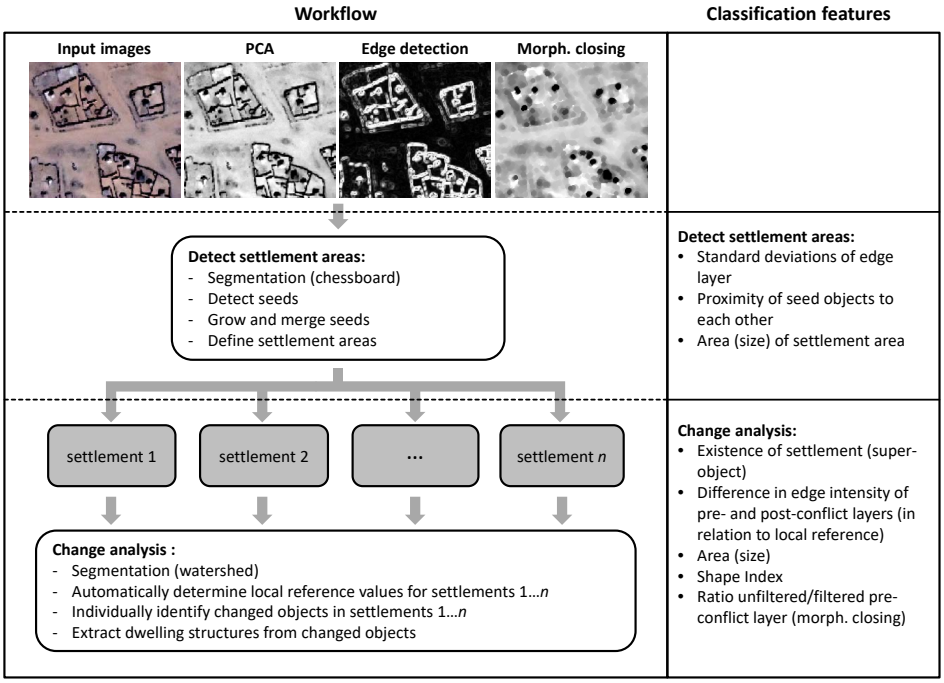

Figure 2. Overview of the example analysis workflow. It shows the key analysis steps on the left and the applied features for the object-based change detection and classification on the right. The first row shows the input images and image processing results created in the first step of the workflow and used as input for the object-based analysis in the subsequent steps (modified after [38]).

2.2.2. Detect Settlement Areas

First, a Principal Component Analysis (PCA) is applied to each file in order to compress the highly redundant spectral information of the three RGB bands to one dimension, the first principal component. The results are then used as pre- and post-conflict layers of a bi-temporal dataset. The detection of settlement areas is performed on a chessboard segmentation level with a 25-m side length. The segments are analyzed regarding the edge intensity of the pre-conflict image since edge detection has proven to be an effective means for the identification and analysis of diverse anthropogenic structures [46–48]. In this example, the edge intensity is determined using an edge detection algorithm after [49]. Segments are analyzed regarding the standard deviation of the edge layer values within each segment. The standard deviation is a good measure to identify anthropogenic structures on this coarse image object level because it accounts for the intensity of edges of those structures in contrast to the background. In addition to the edge intensity, the proximity of segments possibly covering settlement structures (candidate segments) to each other is taken into account. This reflects the specific structure of the sparsely developed villages in Darfur. While dwelling units within these villages are not directly connected, they usually do exist in certain proximity to each other. An area is therefore considered a settlement if it is composed of candidate segments that occur in a certain proximity ($\leq$100 m) to

each other. Those segments are grown and merged into a coherent settlement area. Single candidate segments without proximity to others are ignored.

The threshold for the standard deviation of the edge layer within each segment is a predefined value that can be tuned by the user (see Section 3.2). In addition, a minimum size of a settlement area can be defined to focus the analysis on larger settlements.

2.2.3. Change Analysis within Settlement Areas

The analysis model segments the pre-conflict layer and analyzes the resulting image objects regarding their change values derived from the pre- and post-conflict temporal layers. This exemplary method is an "Image-object overlay", following the categorization by [50]. The workflow can be divided into three major steps: (i) feature extraction and segmentation; (ii) change analysis; and (iii) extraction of dwelling objects from changed objects.

In the first step, image objects are created using a watershed segmentation algorithm, which is based on the identification of local extrema [51]. The segmentation is hampered by the specific structural properties of the objects of interest. The buildings are often directly attached to fences or walls, so they poorly separate from the background. Earlier studies in similar areas showed that the mathematical morphology can eliminate such interfering features [37,52]. Therefore, a morphological closing operator with a disc-shaped structural element of a 3-pixel radius precedes the segmentation to smooth out small and linear features.

In the second step, the objects are analyzed for structural differences between the pre- and post-conflict layer. The analysis focuses on the change of edges because the spectral information in the example images is limited. The change is calculated as the difference in mean edge density per object.

To extract possibly changed objects regarding this change attribute, the applied method does not use absolute values, but instead, detects these objects using a local reference value. Hierarchical relations, using settlement areas as super-objects, allow one to analyze changes of sub-objects with regard to the distribution of the change values of all other sub-objects in the same settlement (e.g., using the mean, minimum or maximum change value). In our example, we compute a "mode value". It represents the most abundant degree of change in the corresponding settlement and is used as a reference. To calculate this reference, the value range of the change feature of all objects within a settlement area is divided into equal intervals. Each object is then classified into its corresponding value range interval. The interval covering the largest area is chosen as the reference interval. The mean change feature value of the objects within the reference interval is defined as the reference value within the corresponding village. All objects are investigated by the difference between their change and the local reference value (for more details, please refer to [38]).

We apply two methods to analyze the objects based on this difference: a thresholding and a k-means cluster analysis. The thresholding defines a minimum value and determines the method's sensitivity to change. The threshold can be tuned manually, and its default value is 0.3. Disappeared objects are detected based on this threshold. The cluster analysis isolates disappeared dwellings in the cluster of highest difference. Its result is not used in the detection of disappeared objects, but provided for manual inspection (see Section 2.2.4).

In the third step, the disappeared objects are investigated regarding their extent, shape and values in the pre-conflict morphological closing layer in relation to the unfiltered layer, i.e., regarding the impact of the closing operator (see Figure A1). The step distinguishes between changed dwelling structures and other, similarly changed objects (e.g., fences). The shape of objects is computed using the shape index [53,54]. It measures how well an object approximates a circle. The more the shape differs from a circle, the higher the shape index value.

The impact of the morphological closing filter on the objects grey scale values is determined to identify features that do not fit the structural element (e.g., small and linear structures), but have similar shape attributes after applying segmentation on the filtered image. This is done by calculating the ratio of the standard deviation values (per object) of the unfiltered pre-conflict layer to those after

filtering. For the shape and size attributes, fixed thresholds are applied to extract objects of interest, i.e., huts and sheds. More details on the object properties and thresholds used in the example analysis can be found in Table A1.

2.2.4. Analysis Output

The workflow produces three outputs (see Figure 3): (i) the main output: a point shapefile with centroids of dwelling objects detected as disappeared; (ii) a polygon shapefile of settlement areas (see Section 2.2.2); and (iii) a supplementary shapefile for understanding the change detection result. It contains polygons of all objects (changed and unchanged) without the thresholding of the second step. The polygon attributes include among others the change cluster (resulting from the unsupervised clustering), as well as the computed change feature for each polygon. They can be used to refine the analysis workflow, e.g., by adapting thresholds. Figure 4 shows outputs (i) disappeared dwellings (resulting from a change sensitivity of 0.33) and (ii) the settlement area (resulting from a settlement detection sensitivity of 0.3). The result well reflects the overall pattern of destruction showing hot spots in the central and southwestern parts of the village. The accuracy is of course impacted by the limited quality and information content of the preview images used as example data. Fifty nine objects were detected as destructed dwelling structures (huts or sheds); 36 of them could be confirmed by visual inspection. Most of the false positives were caused by trees (no infrared channel) and certain configurations of disappeared fences with similar object properties. In addition, we found 20 possibly destructed structures that were not detected by the algorithm.

2.3. Implementation and Packaging of QGIS-Based Workflow

2.3.1. Development in the QGIS Modeler

QGIS is a FOSS GIS [55]. Its processing framework [56] provides access to native QGIS algorithms, as well as a huge number of geoprocessing capabilities of third-party applications, such as GRASS GIS, Orfeo ToolBox, SAGA GIS or R. In addition, user-created algorithms written in Python [57] and consequently the features of any Python library can be added. Analyses can be built using a graphical modeler and are saved as model files. These models can then be run on a selected set of inputs (e.g., layers in the QGIS desktop application) and user-defined parameters.

In our example, the image processing (PCA including subsequent rescaling, morphological closing, edge detection) and segmentation steps are conducted using the algorithms of the Orfeo ToolBox (OTB). OTB is an open-source C++ library for remote sensing and provides a substantial set of image processing tools, including feature extraction, filtering, classification and segmentation algorithms [58]. The segmentation using the OTB process initially creates polygons without any object-specific properties. Therefore, we use native QGIS algorithms to compute image layer value statistics for each of these objects (e.g., mean values regarding edge intensity). Other native algorithms are used for example to perform calculations on object feature values, to extract objects by thresholds or to identify centroids of polygons for the output of results. To compute shape and size properties of objects, the SAGA GIS algorithm polygon shape indices is used [59]. Another SAGA algorithm called identity is applied to establish hierarchical relations: for each object on the level of single buildings (defined as sub-objects), this process identifies the object on the level of settlements (super-objects) in which it is contained. The IDs of these super-objects are saved as attributes of the sub-objects.

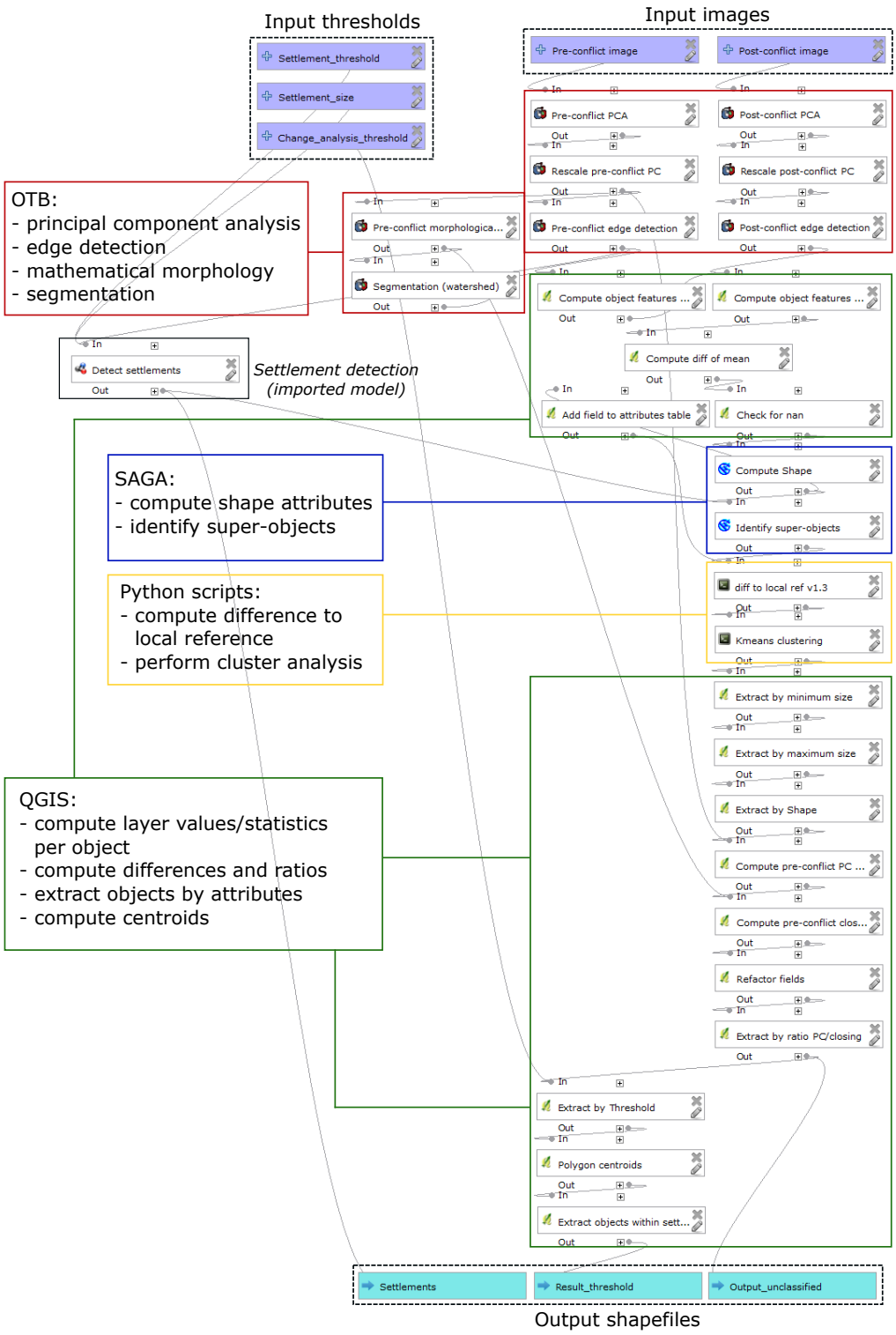

Figure 3. Screenshot of the QGIS graphical workflow modeler showing the example change analysis. Two input images and three numerical parameters are shown in purple boxes at the top. The inputs, processing steps and outputs are connected by grey arcs and roughly ordered from top to bottom. The analysis steps are based on four libraries highlighted by the colored boxes: OTB (red), QGIS (green), SAGA GIS (blue) and Python (yellow). A detailed view of the sub-model detect settlements (grey, left hand side) is in Figure A2. At the bottom, the three output shapefiles are shown in turquoise boxes.

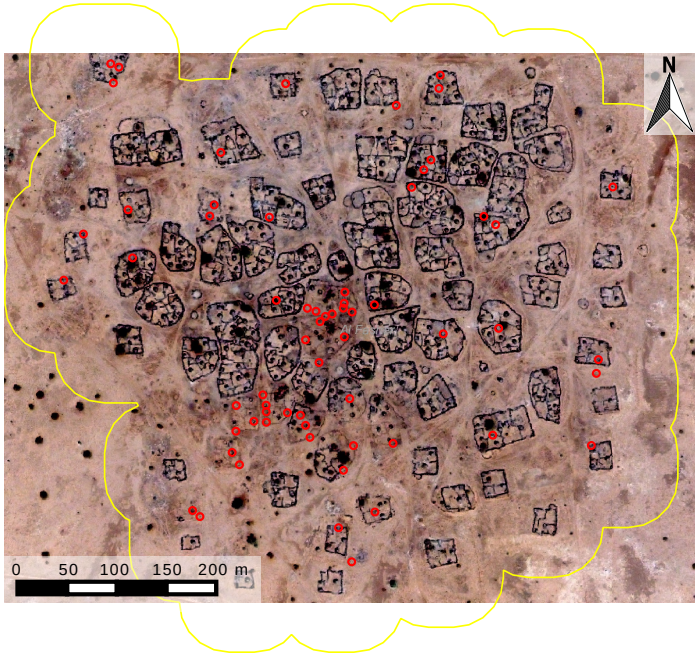

Figure 4. Post-conflict image (location: 13.686°N, 24.979°E) with two results of the example analysis. The detected settlement area is the yellow polygon. The results of the damage assessment, i.e., the disappeared dwellings, are the red circles (image © 2016 DigitalGlobe).

To calculate each sub-object's difference in change to their local reference values (see Section 2.2.3), we developed a Python script. The script uses the super-object IDs to determine sub-objects within the same settlement area and computes the local reference within each settlement individually. It then calculates the difference of each sub-object to the corresponding local reference. For the unsupervised clustering, another Python script was developed because the required algorithm is not available in the QGIS modeler. It reads attribute values of image objects from shapefiles and performs unsupervised clustering of those attributes using the k-means algorithm from the SciPy library [60].

Using the graphical modeler, all processes were combined and saved as a model, which only needs the pre- and post-conflict images as input. Optionally, the thresholds for the settlement detection and the change analysis can be defined (otherwise, the model uses default values). It returns three shapefiles of the three results explained in Section 2.2 as output. Figure 3 shows a screenshot of the modeler view of the change analysis, with annotations of analysis steps and used software packages. A detailed list of all FOSS packages and algorithms used, as well as their function within the workflow can be found in Table A2.

2.3.2. Workspace Preparation

The user workspace comprises the directories and files shown in Listing 1. They are stored in a specific directory structure, so that the model executor can find them. The contents of the workspace are:

- a subdirectory `data` with the two georeferenced data files in TIFF format
- a Python script file, `model.py`, calling the actual model using the QGIS Python API (Application Programming Interface, based on [61])
- analogous to the QGIS models and scripts directories, a `models` and a `scripts` directory containing `.model` (for a visual summary, see Figures 3 and A2, respectively) and Python files

Listing 1: Excerpt of workspace directory tree; the full workspace is available on GitHub [62] and in the reproducibility package, see Section 3.4.

```
/workspace
|-- data
| |-- COPYRIGHT
| |-- jonjona_pos_conflict_proj.tif
| `-- jonjona_pre_conflict_proj.tif
|-- model.py
|-- models
| |-- detect_settlements_on_edgelayer.model
| `-- example_analysis_linux_v3.1.model
`-- scripts
  |-- diff_to_local_ref_v1.3.py
  `-- kmeans_clustering_v2.3.py
```

2.3.3. Containerization of the Workspace and Runtime Environment

The prepared workspace is packaged in an executable container using a tool originally developed for DevOps (cf. [63]) called Docker (http://docker.io). It provides lightweight virtualization and process separation to package an application and its dependencies, for example for scalable deployment in cloud infrastructures. We use a Docker image to encapsulate the GEOBIA workflow with a well-defined software environment. The image can be executed anywhere where a Docker host environment is running, including Linux, Windows and OSX (https://docs.docker.com/engine/installation/).

For execution, a container is started based on an image. A container can be paused, stopped and restarted or be removed from the host. The image is built from a human- and machine-readable definition of the complete environment called `Dockerfile`. This "recipe" allows a scripted definition of Docker images, i.e., installation and configuration of contained software and files, and consequently, a repeatable building of a runtime environment. `Dockerfiles` can start from scratch or a base image and contain arbitrary textual metadata using labels. Image layering allows one to re-use well-vetted images, for example a base image with all typical OBIA software, across projects. The data and specific tools or configuration are added to project-specific images, which can override files and environments of base images.

While not being intended for it, Docker is a means to ensure long-term reproducibility of computational research, as demonstrated for example for R [64]. A Docker image suffices to capture the data, software and runtime environment in a well-defined manner and facilitates reproducibility. For reproduction, a user must only have the concrete project's Docker image. It can be downloaded from an image repository or loaded from a file.

Docker images of software used in our workflow have been published on Docker Hub (for example Todd Stavish's QGIS [65] and OTB [66] images; the Kartoza image for QGIS [67]), but because these execute a GUI by default and do not provide complete control over each software's version, we created our own set of images to run standalone models. Our Dockerfiles are published on GitHub [62] and the corresponding images on Docker Hub (https://hub.docker.com/r/nuest/qgis-model).

Listing 2: Excerpt from the base image Dockerfile. For brevity and illustration, environment variables and commands are shortened.

```
FROM ubuntu:16.04

RUN apt-get update \
  && apt-get install -qqy --no-install-recommends gdal-bin qgis=2.8.6+dfsg-1build1

RUN wget http://[..].sourceforge.net/[..]/saga_2.2.0.tar.gz \
  && tar -xvzf saga*.tar.gz
RUN ./configure && make make install
```

```
RUN wget https://www.orfeo-toolbox.org/[...]/OTB-5.6.1-Linux64.run -q \
  && ./OTB-5.6.1-Linux64.run

ENV PYTHONPATH=/usr/share/qgis/python:/usr/share/qgis/python/plugins
ENV QGIS_WORKSPACE=/workspace
ENV QGIS_MODELFILE=/workspace/models/*.model
ENV QGIS_MODELSCRIPT=/workspace/model.py
ENV QGIS_RESULT=/results
ENV QGIS_USER_MODELDIR=/root/.qgis2/processing/models

WORKDIR /qgis
COPY model.sh model.sh
RUN chmod 0755 model.sh

VOLUME $QGIS_WORKSPACE
VOLUME $QGIS_RESULT

ENTRYPOINT ["/bin/bash", "/qgis/model.sh"]
```

In our specific case, the base image `ubuntu/Dockerfile.xenial` (see Listing 2) installs the required software, sets environment variables and configures the container's default command. The installation commands rely on software packages from the Ubuntu repositories and source installations for SAGA and OTB (SAGA is installed from the source in a specific version not available in the repositories to solve compatibility issues with QGIS; see http://hub.qgis.org/issues/13279 for details). Environment variables provide a single point of configuration. The default command is run with a Bash shell (see [68]). The project Dockerfile `workspace/rs-jonjona/Dockerfile` (see Listing 3) extends the base image by copying the workspace data into the container and by defining an image label with the information about configurable workflow options.

Listing 3: Project Dockerfile.

```
FROM nuest/qgis-model:xenial-multimodel
COPY . /workspace
LABEL de.ifgi.qgis-model.options '[ \
  [...]
  { "id": "change_analysis_threshold", \
   "name": "change sensitivity", \
   "value": "0.3", \
   "comment": "minimum change in edge intensity for objects
        to be flagged as changed" }]'
```

Figure 5 shows the complete control flow in the container. Excerpts from the core files `model.sh` and `model.py` are shown in Listings 4 and 5 respectively (using XVFB [69] for a virtual frame buffer because the container does not have a physical display, but QGIS needs a display even if not used; mounting the hosts physical display would be possible on desktop computers, but not in cloud environments).

Listing 4: Excerpt from `model.sh`; utility code left out for brevity.

```
cp /workspace/models/*.model /root/.qgis2/processing/models
cp /workspace/scripts/*.py /root/.qgis2/processing/scripts
xvfb-run python /workspace/model.py
```

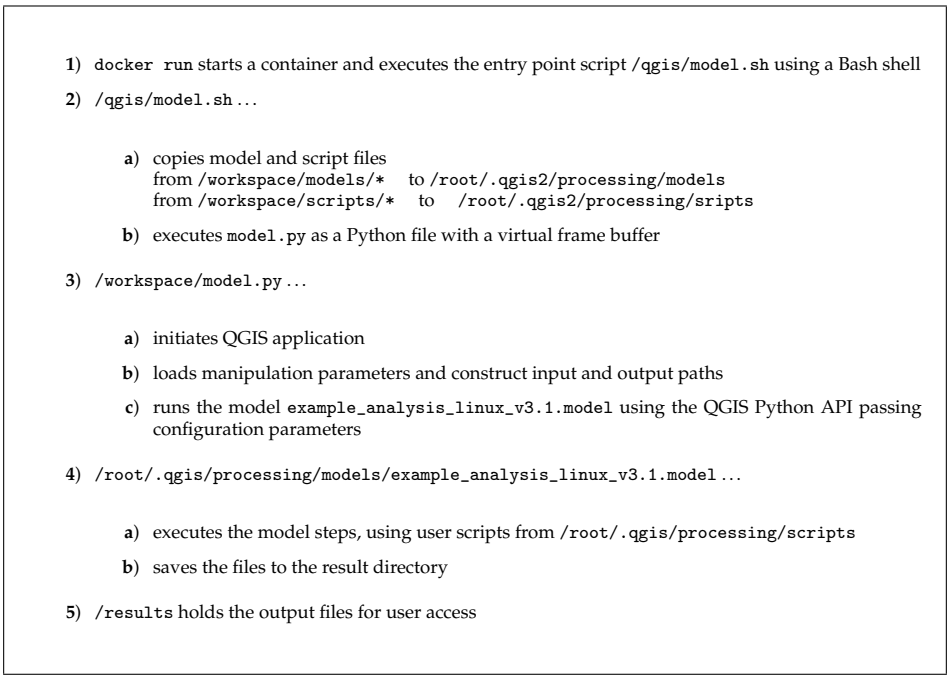

Figure 5. Control flow during an execution of the Docker container in a list of numbered steps. Starting from the command `docker run`, the control flow goes through two script files, one in Bash and one in Python, each in turn acting on other files and being configured using environment variables. Supplementary steps such as logging or loading libraries are omitted for brevity. At the end, the output files are available in a pre-defined directory.

Listing 5: Excerpt from `model.py`; command construction based on environment variables and utility code left out for brevity.

```
app = QgsApplication([], True)
QgsApplication.initQgis()
Processing.initialize()
import processing
processing.runalg("modeler:example_analysis_linux_v3.1", # qgis_model_name
  "/workspace/data/jonjona_pre_conflict_proj.tif",  # inputimage_pre
  "/workspace/data/jonjona_pos_conflict_proj.tif",  # inputimage_post
  0.3,                      # change_analysis_threshold
  0.3,                      # settlement_threshold
  0,                       # settlement_size
  "/results/settlements.shp",           # output_settlements
  "/results/result_threshold.shp",         # output_result_threshold
  "/results/result_unclassified.shp")        # output_result_unclassified
```

2.4. InterIMAGE-Based Analysis

InterIMAGE is another candidate for a FOSS-based OBIA workflow. It provides different segmentation algorithms including the widely-used multiresolution segmentation [70], and operators for calculation of attributes, such as shape, texture or topological characteristics [28]. A so-called batch mode feature has been available since Version 1.39. It allows the automatic execution of InterIMAGE interpretation projects, so-called semantic networks. The networks store the classes and operators to be executed.

We were able to demonstrate running the user interface of the latest available Linux release (1.27) in a Docker container by sharing a local X11 socket [71]. However, several issues hinder the implementation of the use case. Firstly, the software focuses on image interpretation, and not all required algorithms for processing the image layers (e.g., the edge detection) are available in

the basic package. A combination with other tools is possible to add missing functionality (see, e.g., [72,73]), but it is unclear how to achieve that in a scripted workflow. More importantly, the latest available download for Linux is outdated (Version 1.27; see http://www.lvc.ele.puc-rio.br/projects/interimage/download). We were not successful in compiling a later version of the source code for Linux as part of this work due to a lack of documentation and community support (https://groups.google.com/forum/#!topic/interimage/924t-uZrAMs).

While Linux is currently the main operating system for both Docker containers and hosts, support for multi-platform containers exists and is developed further (see [74] for information on Windows containers). Linux containers can be executed in a native Docker for Windows application for recent Windows versions with Hyper-V technology (see https://docs.docker.com/docker-for-windows/). Therefore, Windows-based containers for InterIMAGE will be possible in the future, although the question of licensing is not answered yet.

3. Results

3.1. Running the Container: Command Line Interface

The container can be executed on any computer with Docker. The image with the analysis is published on Docker Hub. Only the first command shown in Listing 6 is required to run the container and reproduce the analysis, because Docker downloads images automatically from Docker Hub. The configuration options enable console output and name the container for later reference. The log (see [75] for a full log) comprises all installed software and their versions, the configured parameters and the output of the started processes.

Listing 6: Full reproduction commands: run the container from Docker Hub and extract the result.

```
docker run -it --name repro nuest/qgis-model:rs-jonjona
docker cp repro:/workspace/results /tmp/repro_results
```

The second command copies the output of the workflow to a directory of the host computer. Listing 7 shows the contents: a directory with a timestamp of the current execution with three shapefiles, the actual model output. The shapefiles can now be inspected or processed further. Figure 4 shows a visualization of the files `result_threshold.shp` and `settlements.shp`.

Listing 7: Result directory tree after execution, supplementary shapefile files, i.e., .dbf, .prj, .qpj, and .shx, and workspace files (see previous Listing 1) not shown.

```
|/result
|‘-- 20161212-172947
|   |-- result_threshold.shp
|   |-- result_unclassified.shp
|   |-- settlements.shp
```

The image can be used to apply the same analysis to another use case. Listing 8 shows the exchange of data (mounting a different workspace) and parameter manipulation (changing the environment variable).

Listing 8: Analysis control and data switching examples. From top to bottom: (a) mounting another workspace; (b) mounting only input files; (c) changing model options via environment variables.

```
# (a)
docker run -it -v /my/analysis:/workspace nuest/qgis-model:rs-jonjona

# (b)
docker run -it -v mypreconflict.tif:/workspace/data/pre_conflict.tif
  -v mypostconflict.tif:/workspace/data/pos_conflict.tif nuest/qgis-model:rs-jonjona

# (c)
docker run -it -e change_analysis_threshold=0.28 nuest/qgis-model:rs-jonjona
```

3.2. Running the Container: Graphical User Interface

As mentioned in Section 1.4, the transferability of rule sets and analysis models is an important aspect in GEOBIA, but some parameters usually need to be adapted to tune an analysis method to a specific study area. In terms of practical application of GEOBIA, a flexible parameterization of analysis workflows by the users must be possible after containerization. Input data need to be easily interchangeable to enable the transfer of a reproducible analysis method to the study area at hand.

To reconcile the issues of reproducibility of and interaction with OBIA workflows, we extended Kitematic (https://kitematic.com/), a FOSS project for managing and running Docker containers with a GUI, with model control functions. We forked the project (https://github.com/nuest/kitematic/tree/model-ui) and added a simple form for controlling the options of workflows (see Figure 6), which hides the complexity of manipulation using environment variables. Instead, the user configures settings (i.e., images to compute the analysis on or thresholds used in the workflow) through fields and buttons. The container's full output log is also readily available.

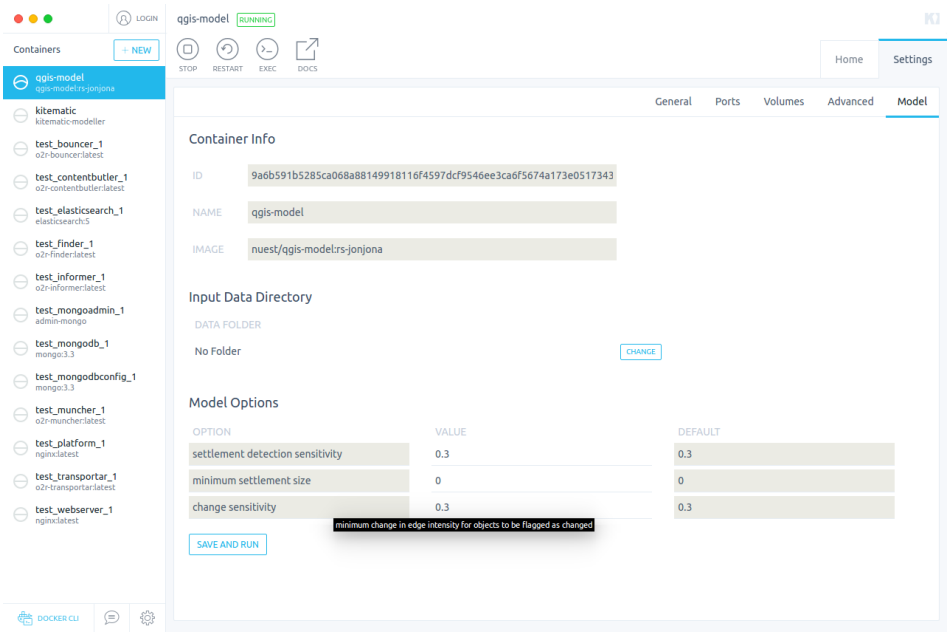

Figure 6. Screenshot of extended Kitematic software. The left-hand side lists locally available containers and the currently-selected one is highlighted in blue. At the top, buttons control the container state. Two nested levels of tabs show information on and allow configuration of the selected container. The tab "model" is active and was developed as part of this work. Its contents in the central area of the UI provide a form-based user interface for controlling parameterized GEOBIA workflows. The list of model options (bottom) shows the option name, current value and default value. A pop-up displays an information text for the third option as the cursor hovers over that line. A "Save and run" button at the bottom can be used to restart the analysis with the changed parameters.

Users can access the result files by mapping the volume where the container stores the results to a directory on the host. Another volume allows one to exchange the whole workspace, i.e., input data and model. Hence, the graphical user interface allows the same level or manipulation as command line options.

3.3. Running InterIMAGE inside Container

Figure 7 shows a screenshot of the InterIMAGE GUI running inside the container. As stated above, it was not possible to run a fully-automated workflow by use of a script as in Section 2.3. Instead,

this example shows a different level of integration, which is to run the user interface of the software inside a container without triggering any predefined models. This provides the users with the full capabilities of that software for building their own analyses without the need to recreate the complete runtime environment.

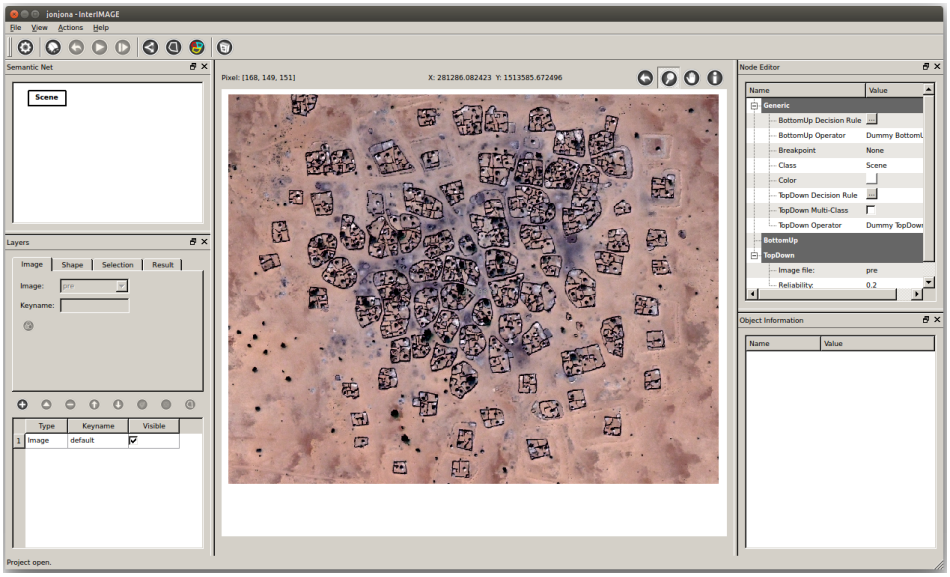

Figure 7. Screenshot of InterIMAGEUI. The software was started with a shared X server using the command `xhost + && docker run -it -rm -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=unix$DISPLAY -e uid=$(id -u) -e gi d=$(id -g) -v /data:/data nuest/docker-interimage:1.27 ./interimage`. The command mounts the display and configures the user within the container to be the same as the executing user. It also mounts a data directory containing the workspace, of which one input image is displayed in the central area of the UI.

3.4. Reproducibility Package

In the spirit of reproducibility and being well-aware of the ephemeral nature of online platforms such as GitHub or Docker Hub, we provide a comprehensive reproducibility package with this work. It contains software, data, Docker images and `Dockerfiles`, documentation of how it was created and instructions on how to run it. It is published at Zenodo, a repository for long-term preservation [75]. The software comprises a current version of Docker and the developed software. Both are included as installers for common operating systems and as source code.

3.5. Reproducible GEOBIA

Three observations can be made for reproducibility in the GEOBIA domain. First, the existing reproducibility spectrum [13] does not represent typical GEOBIA analyzes well because of the limited availability of open data and the dominance of a single commercial software. Since many researchers in the domain actually have access to a de facto standard software, most of them could reproduce an open workspace, although the software is not FOSS. Consequently open-sourcing of the workspace could be distinguished from open-sourcing the used software.

Second, acquisition of remote sensing data is costly, and often, there are no free suitable alternatives. Consequently, one could accept data not being open, because it is readily available for anyone having the financial resources.

Third, no specific guidelines for authors of (GE)OBIA papers exist comparable to the examples from other domains (see Section 1.2). Such guidelines can comprise the aspects documentation, scripted workflows, best practices for project structures, freeware (free as in "free beer"; see [76]) and

FOSS and open data. The former two are already common, yet the latter could have a high impact on reproducibility and practical adoption.

4. Discussion

We successfully demonstrate packaging a complete GEOBIA workflow using FOSS. The package created is transferable between machines (different host operating systems, as well as desktop and cloud platforms), and all tools are available free of charge. This is the first time environment variables and Docker image labels are used to parameterize a scientific workflow with a GUI. Our experiments show that containerization is useful not only for reproducibility by third parties, but also for the original development of a FOSS-based analysis, because of the numerous tools involved in different versions and potential conflicts between them. The customized user interface removes barriers for practitioners with limited computer science experience. However, this only concerns the use of a containerized method. For authors of new methods, the creation of a container requires knowledge in the area of Docker and the QGIS Python API.

A Docker container is not a black box, since each applied software is documented in detail in the Docker file. The presented solution also allows one to customize a method by changing input parameters and data. In our example, we only enable three parameters to be manipulated, but the approach can accommodate any number of additional variables (e.g., segmentation parameters). It is also possible to develop more complex containers allowing users to choose different algorithms. We thereby present a means to technically reconcile the conflicting priorities of customizability and reproducibility. However, the conceptual question on the desired degree of customizability, i.e., to what extent the target group of practitioners is expected to redevelop a methodology provided by an expert, remains open and strongly depends on the specific use case.

Our work also reveals challenges with regard to the overarching goal of reproducible research. The presented solution is arguably a one-off effort to containerize a specific workflow and does not require any standardization beyond the `docker run` command. Yet, only an experienced developer can trace the complete flow of information, from the Docker entry point via used scripts to the actually executed code and used parameters, to grasp the complete picture. The user scripts and analysis model are embedded in the container, and extracting them requires the container to be started. This could be a barrier for users and for the purposes of development and exploration, but a copy could be kept outside the image or be made available by reproducibility tools. Keeping a copy outside the image naturally leads to a nested packaging approach.

The selection of Docker as the container engine makes it crucial for both reproduction and archiving. This dependency on a specific product is mitigated by Docker being open-source and highly adopted in the IT industry. An open standardization effort also is underway: the Open Container Initiative (https://www.opencontainers.org/). This effort contributes to a proper long-term archiving solution because the runtime environment must be preserved, and archiving cannot rely solely on the `Dockerfile` without replicating all source repositories or download sites. Layers of images, i.e., base images and an analysis image, make it possible to easily store, share, adopt and collaborate on complex analyses because they can be shared or extended further. However, they also increase complexity.

The approach for UI-based GEOBIA within containers (see Section 3.3) using the current tools is only possible on Unix-based operating systems. Interactive interfaces for containerized workflows are possible in an OS-independent manner by developing the whole analysis within a container, which provides a user interface via HTTP and HTML to a regular web browser (cf. [77]). The advantages are complete and consistent containers and immediate visual access to results. However, we suspect that most users prefer to develop an analysis in the environment they are used to and only package a complete analysis.

Besides the technical challenges, best practices for reproducible research could provide a meaningful yet generic workspace structure and enforce general practices, such as managing scripts in a version control system [78]. Such practices could be implemented in a standardized format

for container-based reproducibility packages, be supported by ready-to-use templates and even be partially automated for example with an "export to container"-button in the QGIS workflow modeler to generate a `Dockerfile`. The presented solution can accommodate such best practices. The formal specification of a container format and supporting services for semi-automatic creation are subject of current research and can mitigate the above-mentioned knowledge requirements for the authors of methods (cf. [79]).

The availability of open data remains a general issue. Especially in GEOBIA, where very high resolution imagery plays an important role in many analyses, the applied images are often not freely available. In these cases, it is not possible to publish the data along with the analysis workflow and software. On the other hand, GEOBIA is a widely-used tool in the rapidly-growing field of analyzing very high-resolution data, e.g., from unmanned aerial systems (UAS). Here, scientists become producers of their own image data. This makes an approach as presented here especially useful because the studied images can be published and at the same time become an essential requirement for full reproducibility due to their uniqueness.

The FOSS solutions applied in the containerized workflow, at the current stage, cannot compete with commercial software packages, such as eCognition, regarding functionality and data models for GEOBIA. The available functions of FOSS tools already provide a substantial set of algorithms, and the analysis is created with a user-friendly interactive modeler in a Desktop environment. However, the number of actual OBIA operations for image interpretation is limited. Our example analysis shows that many aspects of GEOBIA can already be realized, e.g., by a combination of algorithms and tailored scripts. However, more complex models, including iterative sequences of segmentation, merging and interpretation of objects (e.g., for a better extraction of relevant dwelling structures, cf. [46,80]) are still difficult to develop in FOSS. However, since FOSS tools are easily extensible, the missing functionality can be contributed as new functions or independent tools.

5. Conclusions

Docker containers and a combination of established free and open-source GIS and image analysis software enable reproducible GEOBIA. We build and distribute a container to carry all required software and data in a transparent manner. The provided user interface makes the package easy to use. This is a breakthrough for creating a transferable and executable package of a GEOBIA workflow. The presented analysis goes well beyond simple processing by successfully integrating tools into a complex multi-step analysis. Packaging GEOBIA software and workflows opens new possibilities for reviews of scientific work, collaboration between researchers and adoption by practitioners. The example analysis in conflict damage assessment presents an application field where transparency and cost are important factors, so that an open approach is advantageous. The shortcomings with respect to the reproducibility of analyses are mostly related to usability. To reach a comprehensive feature set, high user-friendliness and subsequently practical adoption, a community of GEOBIA users applying and contributing to open-source technologies is needed. Although there are commonalities across all scientific disciplines, domain-specific requirements demand: (i) targeted education; (ii) high-quality specialized FOSS; and (iii) best practices. The challenge starts with an open discourse on reproducible research and a working definition of reproducibility specifically for GEOBIA (cf. [81]), to which this work intends to be a first step.

Acknowledgments: This research has been conducted in the context of the Graduate School for Geoinformatics (http://www.uni-muenster.de/Geoinformatics/en/Studies/study_programs/PhD/). It has partly been supported by the project Opening Reproducible Research (http://o2r.info and https://www.uni-muenster.de/forschungaz/project/9520) funded by the German Research Foundation (DFG) under Project Number PE 1632/10-1. We thank Edzer Pebesma for valuable comments and support.

Author Contributions: Both authors contributed equally to the paper. Christian Knoth conceived the study and developed the analysis workflow and its FOSS-based implementation. Daniel Nüst performed the containerization of the workflow and runtime environment and implemented the user interface for the interaction with the containerized workflow.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
GIS	Geographic Information System
GEOBIA	Geographic Object-Based Image Analysis
OBIA	Object-Based Image Analysis
FOSS	Free and Open-Source Software
LULC	Land Use and Land Cover
LIDAR	Light Detection and Ranging
GUI	Graphical User Interface
AAAS	American Association for the Advancement of Science
PCA	Principal Component Analysis
OTB	Orfeo ToolBox
SAGA	System for Automated Geoscientific Analyses
API	Application Programming Interface
XVFB	X Window Virtual Frame Buffer
UAS	Unmanned Aerial Systems
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language

Appendix A

Table A1. List of properties used in the settlement detection and the subsequent detection of disappeared structures (within detected settlement areas) along with the corresponding rules and recommended thresholds.

Object Property	Rule or Threshold	Analysis Step
Standard deviation of edge layer (pre-conflict) of seed segments	≥ 0.3	Settlement detection
Proximity of seed segments to each other	≤ 100 m	Settlement detection
Number of seed segments (per settlement)	≥ 2	Settlement detection
Optionally: Size of settlement area (after merging of seeds)	≥ 0 (no default threshold set in this example)	Settlement detection
Existence of super-object of class settlement	True (super-object ID > 0)	Change analysis
Change of edge intensity	Difference to local reference value ≥ 0.33	Change analysis
Minimum size (area)	10 m^2	Change analysis
Maximum size (area)	60 m^2	Change analysis
Shape Index value	≤ 1.55	Change analysis
Impact of morphological closing (ratio of standard deviation of pre-conflict layer values per object before and after morphological closing)	≤ 5.5	Change analysis

Table A2. Summary of the QGIS-based analysis workflow showing the processing steps and the corresponding algorithms (Python: the scripts written in Python; see Section 2.3.1).

Analysis Step	Algorithm	Stage of Workflow
Extract first principal component of pre- and post-conflict image	OTB:DimensionalityReduction (pca)	Image processing
Rescale both principal components to 8bit	OTB:Rescale Image	Image processing
Edge detection on both layers	OTB:EdgeExtraction (touzi)	Image processing
Morphological closing on pre-conflict layer	OTB:GrayScaleMorphologicalOperation (closing)	Image processing
Determine extent of raster layer	QGIS:Raster layer bounds	Settlement detection
Create chessboard segmentation within extent	QGIS:Create grid	Settlement detection
Compute standard deviation of edge layer within segments	QGIS:Zonal statistics	Settlement detection
Extract settlement candidate segments according to standard deviation of edge layer	QGIS:Extract by attribute	Settlement detection
Create settlement area objects by growing and merging candidate segments that are within proximity (100 m max) to each other (ignore isolated candidates)	QGIS: Fixed distance buffer QGIS:Multipart to singleparts SAGA:Polygon shape indices QGIS:Extract by attribute QGIS:Fill holes	Settlement detection
Create IDs in attribute table and specify field name	QGIS:Add autoincremental field QGIS:Refactor fields	Settlement detection
Create objects on level of single huts	OTB:Segmentation (watershed)	Change analysis
Compute mean of edge intensity within objects (pre- and post-conflict)	QGIS:Zonal statistics	Change analysis
Calculate difference in mean edge density between pre- and post-conflict (check for NULL)	QGIS:Adv. Python Field Calculator QGIS:Extract by attribute	Change analysis
Compute shape and size properties of objects	SAGA:Polygon shape indices	Change analysis
For all sub-objects, get IDs of containing super-objects (settlements)	SAGA:Identity	Change analysis
Compute local reference (of change) within settlements and difference of sub-objects to this reference	Python:Difference to local reference v1.3	Change analysis
Compute unsupervised clustering regarding change	Python:Kmeans clustering v2.3	Change analysis
Extract objects by minimum and maximum size	QGIS:Extract by attribute	Change analysis
Extract objects by their shape index	QGIS:Extract by attribute	Change analysis
Compute statistics of pre-conflict layer per object before and after morphological closing	QGIS:Zonal statistics	Change analysis
Calculate ratio of sdev. values of pre-conflict layer before and after morphological closing	QGIS:Refactor fields	Change analysis
Extract objects by ratio value	QGIS:Extract by attribute	Change analysis
Extract objects by change value (difference in mean edge density) using pre-defined threshold	QGIS:Extract by attribute	Change analysis
Compute centroids of objects extracted by threshold and within settlements	QGIS:Polygon centroids QGIS:Extract by attribute	Change analysis

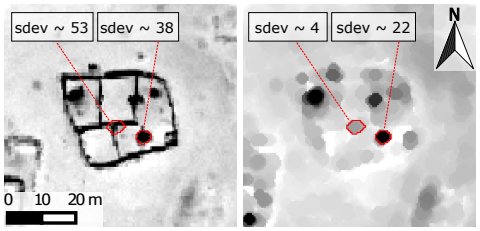

Figure A1. Subset of the pre-conflict layer before (left image) and after (right image) morphological closing. It shows the effect of the filter on a dwelling object (right object) and a fence (left object). The higher impact of the filter on small, linear structures is used as an additional feature to remove them by measuring the ratio of the standard deviation per object of the unfiltered to that of the filtered layer.

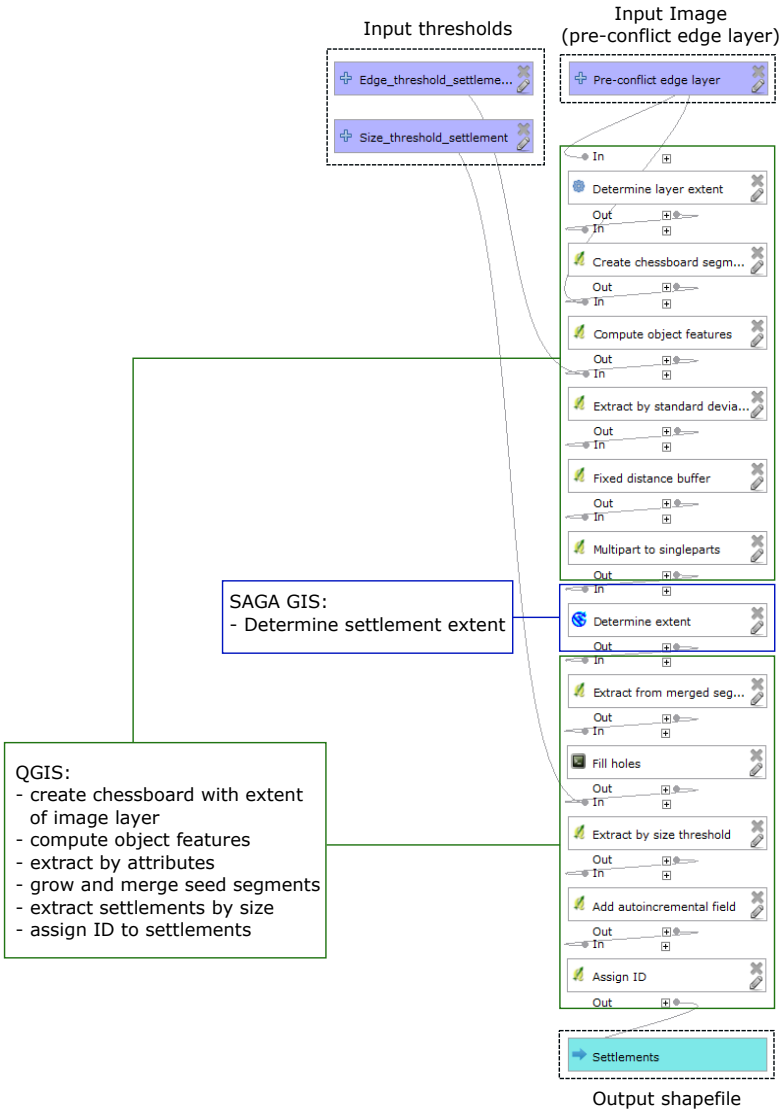

Figure A2. Screen-shot of analysis workflow in the QGIS graphical modeler with highlighting. Three inputs, two numerical thresholds and the pre-conflict edge layer are shown in purple at the top. The analysis steps are connected with grey arcs and executed from top to bottom. They are based on QGIS (green boxes) and SAGA GIS (dark blue box). This model is applied as one algorithm in the analysis workflow depicted in Figure 3. The model output is a single shape file with detected settlements, shown in turquoise at the bottom.

References

1. Bailey, C.W. What Is Open Access? Available online: http://digital-scholarship.org/cwb/WhatIsOA.htm (accessed on 19 December 2016).
2. European Commission Horizon 2020 Open Science (Open Access) Available online: https://ec.europa.eu/programmes/horizon2020/en/h2020-section/open-science-open-access (accessed on 19 December 2016).
3. The Commission High Level Expert Group on the European Open Science Cloud *Realising the European Open Science Cloud*; European Commission: Brussels, Belgium, 2016.
4. Nosek, B.A.; Alter, G.; Banks, G.C.; Borsboom, D.; Bowman, S.D.; Breckler, S.J.; Buck, S.; Chambers, C.D.; Chin, G.; Christensen, G.; et al. Promoting an open research culture. *Science* **2015**, *348*, 1422–1425.
5. Peng, R.D. Reproducible research and Biostatistics. *Biostatistics* **2009**, *10*, 405–408.
6. Markowetz, F. Five selfish reasons to work reproducibly. *Genome Biol.* **2015**, *16*, 274.
7. Kraker, P.; Dörler, D.; Ferus, A.; Gutounig, R.; Heigl, F.; Kaier, C.; Rieck, K.; Šimukovič, E.; Vignoli, M.; Aspöck, E.; et al. The Vienna Principles: A Vision for Scholarly Communication in the 21st Century. *Mitteilungen der Vereinigung Österreichischer Bibliothekarinnen und Bibliothekare* **2016**, *3*, 436–446.
8. Sandve, G.K.; Nekrutenko, A.; Taylor, J.; Hovig, E. Ten Simple Rules for Reproducible Computational Research. *PLoS Comput. Biol.* **2013**, *9*, e1003285.
9. Gentleman, R.; Temple Lang, D. Statistical Analyses and Reproducible Research. *J. Comput. Graph. Stat.* **2007**, *16*, 1–23.
10. Goodman, S.N.; Fanelli, D.; Ioannidis, J.P.A. What does research reproducibility mean? *Sci. Transl. Med.* **2016**, *8*, 341ps12.
11. Amitrano, D.; Di Martino, G.; Iodice, A.; Riccio, D.; Ruello, G. A New Framework for SAR Multitemporal Data RGB Representation: Rationale and Products. *IEEE Trans. Geosci. Remote Sens.* **2015**, *53*, 117–133.
12. Howe, B. Virtual Appliances, Cloud Computing, and Reproducible Research. *Comput. Sci. Eng.* **2012**, *14*, 36–41.
13. Peng, R.D. Reproducible Research in Computational Science. *Science* **2011**, *334*, 1226–1227.
14. Lucieer, V.; Hill, N.A.; Barrett, N.S.; Nichol, S. Do marine substrates 'look' and 'sound' the same? Supervised classification of multibeam acoustic data using autonomous underwater vehicle images. *Estuar. Coast. Shelf Sci.* **2013**, *117*, 94–106.
15. Dupuy, S.; Barbe, E.; Balestrat, M. An Object-Based Image Analysis Method for Monitoring Land Conversion by Artificial Sprawl Use of RapidEye and IRS Data. *Remote Sens.* **2012**, *4*, 404–423.
16. Tormos, T.; Dupuy, S.; Van Looy, K.; Barbe, E.; Kosuth, P. An OBIA for fine-scale land cover spatial analysis over broad territories: Demonstration through riparian corridor and artificial sprawl studies in France. In Proceedings of the GEOBIA 2012: 4th International Conference on GEographic Object-Based Image Analysis, Rio de Janeiro, Brazil, 7–9 May 2012.
17. Stodden, V.; Miguez, S.; Seiler, J. ResearchCompendia.org: Cyberinfrastructure for Reproducibility and Collaboration in Computational Science. *Comput. Sci. Eng.* **2015**, *17*, 12–19.
18. Bechhofer, S.; Buchan, I.; De Roure, D.; Missier, P.; Ainsworth, J.; Bhagat, J.; Couch, P.; Cruickshank, D.; Delderfield, M.; Dunlop, I.; et al. Why linked data is not enough for scientists. *Future Gen. Comput. Syst.* **2013**, *29*, 599–611.
19. Chirigati, F.; Rampin, R.; Shasha, D.; Freire, J. ReproZip: Computational Reproducibility With Ease. In Proceedings of the SIGMOD 2016 International Conference on Management of Data, San Francisco, CA, USA, 26 June–1 July 2016.
20. Douglas T.; Haiyan Meng, P. Techniques for Preserving Scientific Software Executions: Preserve the Mess or Encourage Cleanliness? In Proceedings of the 12th International Conference on Digital Preservation (iPres) 2015, Chapel Hill, NC, USA, 2–6 November 2016.
21. Open Source Initiative. Open Source Case for Business:Advocacy. Available online: https://opensource.org/advocacy/case_for_business.php (accessed on 19 December 2016).
22. Wightman, T.. What's keeping you from using open source software?. Available online: https://opensource.com/business/13/12/using-open-source-software (accessed on 19 December 2016).
23. Salus, P. *A Quarter-Century of Unix*; Addison-Wesley: Boston, MA, USA, 1994; pp. 52–53.

24. Grippa, T.; Lennert, M.; Beaumont, B.; Vanhuysse, S.; Stephenne, N.; Wolff, E. An Open-Source Semi-Automated Processing Chain for Urban OBIA Classification. In Proceedings of the GEOBIA 2016: Solutions & Synergies, Enschede, The Netherlands, 14–16 September 2016.
25. Böck, S.; Immitzer, M.; Atzberger, C. Automated Segmentation Parameter Selection and Classification of Urban Scenes Using Open-Source Software. In Proceedings of the GEOBIA 2016: Solutions & Synergies, Enschede, The Netherlands, 14–16 September 2016.
26. Van De Kerchove, R.; Hanson, E.; Wolff, E. Comparing pixel-based and object-based classification methodologies for mapping impervious surfaces in Wallonia using ortho-imagery and LIDAR data. In Proceedings of the GEOBIA 2014: Advancements, Trends and Challenges, Thessaloniki, Greece, 22 May 2014.
27. Körting, T.; Fonseca, L.; Câmara, G. GeoDMA—Geographic Data Mining Analyst. *Comput. Geosci.* **2013**, *57*, 133–145.
28. Costa, G.; Feitosa, R.; Fonseca, L.; Oliveira, D.; Ferreira, R.; Castejon, E. Knowledge-based interpretation of remote sensing data with the InterImage system: Major characteristics and recent developments. In Proceedings of the GEOBIA 2010: Geographic Object-Based Image Analysis, Ghent, Belgium, 29 June–2 July 2010.
29. Antunes, R.; Happ, P.; Bias, E.; Brites, R.; Costa, G.; Feitosa, R. An Object-Based Image Interpretation Application on Cloud Computing Infrastructure. In Proceedings of the GEOBIA 2016: Solutions & Synergies, Enschede, The Netherlands, 14–16 September 2016.
30. Blaschke, T.; Hay, G.; Maggi, K.; Lang, S.; Hofmann, P.; Addink, E.; Feitosa, R.; van der Meer, F.; van der Werff, H.; van Coillie, F.; et al. Geographic Object-Based Image Analysis—Towards a new paradigm. *ISPRS J. Photogramm. Remote Sens.* **2014**, *87*, 180–191.
31. Drăguţ, L.; Tiede, D.; Levick, S. ESP: A tool to estimate scale parameter for multiresolution image segmentation of remotely sensed data. *Int. J. Geograph. Inf. Sci.* **2010**, *24*, 859–871.
32. Drăguţ, L.; Csillik, O.; Eisank, C.; Tiede, D. Automated parameterisation for multi-scale image segmentation on multiple layers. *ISPRS J. Photogramm. Remote Sens.* **2014**, *88*, 119–127.
33. Martha, T.; Kerle, N.; van Westen, C. Segment Optimization and Data-Driven Thresholding for Knowledge-Based Landslide Detection by Object-Based Image Analysis. *Int. J. Geograph. Inf. Sci.* **2011**, *49*, 4928–4943.
34. Hofmann, P. Defuzzification Strategies for Fuzzy Classifications of Remote Sensing Data. *Remote Sens.* **2016**, *8*, 467.
35. Kohli, D.; Warwadekar, P.; Kerle, N.; Sliuzas, R.; Stein, A. Transferability of Object-Oriented Image Analysis Methods for Slum Identification. *Remote Sens.* **2013**, *5*, 4209–4228.
36. Hofmann, P.; Blaschke, T.; Strobl, J. Quantifying the robustness of fuzzy rule sets in object-based image analysis. *Int. J. Remote Sens.* **2011**, *32*, 7359–7381.
37. Knoth, C.; Pebesma, E. Detecting destruction in conflict areas in Darfur. In Proceedings of the GEOBIA 2014: Advancements, Trends and Challenges, Thessaloniki, Greece, 22 May 2014.
38. Knoth, C.; Pebesma, E. Detecting dwelling destruction in Darfur through object-based change analysis of very high-resolution imagery. *Int. J. Remote Sens.* **2017**, *38*, 273–295.
39. Tiede, D.; Füreder, P.; Lang, S.; Hölbling, D.; Zeil, P. Automated Analysis of Satellite Imagery to provide Information Products for Humanitarian Relief Operations in Refugee Camps - from Scientific Development towards Operational Services. *Photogramm. Fernerkund. Geoinf.* **2013**, *2013*, 185–195.
40. Giada, S.; De Groeve, T.; Ehrlich, D.; Soille, P. Information extraction from very high resolution satellite imagery over Lukole refugee camp, Tanzania. *Int. J. Remote Sens.* **2003**, *24*, 4251–4266.
41. Al-Khudhairy, D.; Caravaggi, I.; Giada, S. Structural damage assessments from Ikonos data using change detection, Object-level Segmentation, and Classification Techniques. *Photogramm. Eng. Remote Sens.* **2005**, *71*, 825–837.
42. Witmer, F. Remote sensing of violent conflict: Eyes from above. *Int. J. Remote Sens.* **2015**, *36*, 2326–2352.
43. Wolfinbarger, S. Remote Sensing as a Tool for Human Rights Fact-Finding. In *The Transformation of Human Rights Fact-Finding*; Alston, P.; Knuckey, S., Eds.; Oxford University Press: New York, NY, USA, 2016; pp. 463–477.
44. Knoth, C.; Nüst, D. Enabling reproducible OBIA with open-source software in docker containers. In Proceedings of the GEOBIA 2016: Solutions & Synergies, Enschede, The Netherlands,

14–16 September 2016; Kerle, N., Gerke, M., Lefèvre, S., Eds.; University of Twente Faculty of Geo-Information and Earth Observation (ITC): Enschede, The Netherlands, 2016.
45. American Association for the Advancement of Science Appendix A: Darfur, Sudan and Chad Imagery Characteristics Available online: http://www.aaas.org/page/appendix-darfur-sudan-and-chad-imagery-characteristics (accessed on 19 December 2016).
46. Lang, S.T.D.; Hölbling, D.; Füreder, P.; Zeil, P. Earth observation (EO)-based ex post assessment of internally displaced person (IDP) camp evolution and population dynamics in Zam Zam, Darfur. *Int. J. Remote Sens.* **2010**, *31*, 5709–5731.
47. Wei, Y.; Zhao, Z.; Song, J. Urban building extraction from high-resolution satellite panchromatic image using clustering and edge detection. In Proceedings of the 2004 IEEE International Geoscience and Remote Sensing Symposium, Anchorage, AK, USA, 20–24 September 2004.
48. De Kok, R.; Wezyk, P. Principles of full autonomy in image interpretation. The basic architectural design for a sequential process with image objects. In *Object-Based Image Analysis. Spatial Concepts for Knowledge-Driven Remote Sensing Applications*; Blaschke, T.; Lang, S.; Hay, G.J., Eds.; Springer: Berlin, Germany, 2008; pp. 697–710.
49. Touzi, R.; Lopes, A.; Bousquet, P. A statistical and geometrical edge detector for SAR images. *IEEE Trans. Geosci. Remote Sens.* **1988**, *26*, 764–773.
50. Tewkesbury, A.P.; Comber, A.J.; Tate, N.J.; Lamb, A.; Fisher, P.F. A critical synthesis of remotely sensed optical image change detection techniques. *Remote Sens. Environ.* **2015**, *160*, 1–14.
51. OTB Development Team. The ORFEO Tool Box Software Guide. Available online: https://www.orfeo-toolbox.org//packages/OTBSoftwareGuide.pdf (accesssed on 27 June 2016).
52. Sulik, J.; Edwards, S. Feature extraction for Darfur: Geospatial applications in the documentation of human rights abuses. *Int. J. Remote Sens.* **2010**, *31*, 2521–2533.
53. Lang, S.; Blaschke, T. *Landschaftsanalyse Mit GIS*; Ulmer: Stuttgart, Germany, 2007; pp. 241–243.
54. Forman, R.; Godron, M. *Landscape Ecology*; Wiley: New York, NY, USA, 1986; pp. 106–108.
55. QGIS Development Team. QGIS Geographic Information System. Available online: http://qgis.osgeo.org (accessed on 24 June 2016).
56. Graser, A.; Oyala, V. Processing: A Python Framework for the Seamless Integration of Geoprocessing Tools in QGIS. *ISPRS Int. J. Geo-Inf.* **2015**, *4*, 2219–2245.
57. Rossum, G. Python Reference Manual. Available online: http://www.python.org/ (accessed on 19 December 2016).
58. Inglada, J.; Christophe, E. The Orfeo Toolbox remote sensing image processing software. In Proceedings of the 2009 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Cape Town, South Africa, 12–17 July 2009.
59. Conrad, O.; Bechtel, B.; Bock, M.; Dietrich, H.; Fischer, E.; Gerlitz, L.; Wehberg, J.; Wichmann, V.; Böhner, J. System for Automated Geoscientific Analyses (SAGA) v. 2.1.4. *Geosci. Model Dev.* **2015**, *8*, 1991–2007.
60. Jones, E.; Oliphant, T.; Peterson, P. SciPy: Open Source Scientific Tools for Python. Available online: http://www.scipy.org/ (accessed on 27 June 2016).
61. PyQGIS Developer Cookbook Using PyQGIS in standalone scripts. Available online: http://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/intro.html#using-pyqgis-in-standalone-scripts (accessed on 27 June 2016).
62. Nüst, D. Docker Container for QGIS Models on GitHub. Available online: https://github.com/nuest/docker-qgis-model (accessed on 24 November 2016).
63. Loukides, M. *What is DevOps? Infrastructure as Code*; O'Reilly Media: Sebastopol, CA, USA, 2012.
64. Boettiger, C. An introduction to Docker for reproducible research, with examples from the R environment. *ACM SIGOPS Oper. Syst. Rev.* **2015**, *49*, 71–79.
65. Stavish, T. Docker for QGIS. Available online: https://hub.docker.com/r/toddstavish/qgis (accessed on 24 November 2016).
66. Stavish, T. Docker for OTB. Available online: https://hub.docker.com/r/toddstavish/orfeo_toolbox (accessed on 24 November 2016).
67. Sutton, T. Docker for QGIS Desktop. Available online: https://hub.docker.com/r/kartoza/qgis-desktop (accessed on 24 November 2016).

68. GNU-Project GNU Bash Available online: https://www.gnu.org/software/bash/ (accessed on 24 November 2016).
69. Wiggins, D.P. XVFB Documentation. Available online: https://www.x.org/releases/X11R7.6/doc/man/man1/Xvfb.1.xhtml (accessed on 24 November 2016).
70. Baatz, M.; Schäpe, A. Multiresolution segmentation—An optimization approach for high quality multi-scale image segmentation. In *Angewandte Geographische Informations-Verarbeitung XII*; Strobl, J.; Blaschke, T.; Griesebner, G., Eds.; Wichmann: Karlsruhe, Germany, 2000; pp. 12–23.
71. Nüst, D.; Knoth, C. Docker-Interimage: Running the Latest InterIMAGE Linux Release in A Docker Container with User Interface. Available online: http://zenodo.org/record/55083 (accessed on 6 July 2016).
72. Antunes, R.; Bias, E.; Brites, R.; Costa, G. Integration of Open-Source Tools for Object-Based Monitoring of Urban Targets. In Proceedings of the GEOBIA 2016: Solutions & Synergies, Enschede, The Netherlands, 14–16 September 2016.
73. Passo, D.; Bias, E.; Brites, R.; Costa, G.; Antunes, R. Susceptibility mapping of linear erosion processes using object-based analysis of VHR images. In Proceedings of the GEOBIA 2016: Solutions & Synergies, Enschede, The Netherlands, 14–16 September 2016.
74. Friis, M. Docker on Windows Server 2016 Technical Preview 5. Available online: https://blog.docker.com/2016/04/docker-windows-server-tp5/ (accessed on 19 December 2016)
75. Nüst, D.; Knoth, C. Data and code for: Reproducibility and Practical Adoption of GEOBIA with Open-Source Software in Docker Containers. Available online: https://doi.org/10.5281/zenodo.168370 (accessed on 19 December 2016).
76. GNU-Project What is free software? Available online: https://www.gnu.org/philosophy/free-sw.en.html (accessed on 24 November 2016).
77. Marwick, B. 1989-Excavation-Report-Madjebebe. Available online: https://doi.org/10.6084/m9.figshare.1297059.v2 (accessed on 24 November 2016).
78. Ram, K. Git can facilitate greater reproducibility and increased transparency in science. *Source Code Biol. Med.* **2013**, *8*, 7.
79. Nüst, D.; Konkol, M.; Schutzeichel, M.; Pebesma, E.; Kray, C.; Przibytzin, H.; Lorenz, J. Opening the Publication Process with Executable Research Compendia. *D-Lib Mag.* **2017**, doi:10.1045/january2017-nuest.
80. Tiede, D.; Lang, S.; Hölbling, D.; Füreder, P. Transferability of OBIA rulesets for IDP camp analysis in Darfur. In Proceedings of the GEOBIA 2010: Geographic Object-Based Image Analysis, Ghent, Belgium, 29 June–2 July 2010.
81. Baker, M. Muddled meanings hamper efforts to fix reproducibility crisis. *Nat. News* **2016**, doi:10.1038/nature.2016.20076.