Master Thesis

submitted within the UNIGIS MSc programme
Interfaculty Department of Geoinformatics - Z_GIS
University of Salzburg

Assess the level of damage due to a flood event using a building footprint

by

René Kaspar

105166, UNIGIS MSc 2020

A thesis submitted in partial fulfilment of the requirements of
the degree of
Master of Science – MSc

Advisor:

Assoz-Prof. Dr. Stefan Lang

Zürich, 29.09.2021

Erklärung der eigenständigen Abfassung der Arbeit

Ich versichere, diese Master Thesis ohne fremde Hilfe und ohne Verwendung anderer als der angeführten Quellen angefertigt zu haben, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen ist. Alle Ausführungen der Arbeit, die wörtlich oder sinngemäss übernommen wurden, sind entsprechend gekennzeichnet.

Declaration of Authorship

I hereby declare that this thesis is my own work and I have not used any sources or aids other than the ones cited. This thesis has not been presented in the same or similar form to another examination board. All statements in this thesis that have been quoted direct or indirect are marked.

Zürich, 29 September 2021

René Kaspar

Table of Contents

Abstract

Flood mapping and damage assessments with remote sensing data became an effective instrument over the last decade, as it allows fast first evaluation of flood events. However, research in low-cost and effective flood event evaluation methods for developing countries is limited. More accurate information in regards to flood events would allow flood relief management to be more precise and efficient. A helpful proxy information to assess the number of affected people and the property damage is affected building structures. Through automatic extraction of building footprints from satellite images, vector data for building structures are now freely available for some countries through open-source initiatives. For example, in 2019 Microsoft published a country wide Building Footprint for Tanzania and Uganda. In this thesis, the intention is to figure out, if the open-source Building Footprint can be reliably used to create a low-cost method to assess the number of affected buildings during a flood event.

Since automatically extracted building footprint data at this scale and accuracy, is relatively new, its usability in this kind of damage evaluation needs to be assessed to determine the accuracy of the results. The city of Dar es Salaam was selected as a study site as there is high quality Open Street Map (OSM) data available to verify the results. In order to assess the damage, two indicators were defined, number of affected buildings and total area covered by the affected buildings. To visualize the spatial distribution of the affected buildings, maps were also generated. Assessment of the indicators and visualization were done for both, Building Footprint and the OSM data, for comparison. An automatic workflow was established to calculate these indicators and generate the map to ensure the workflow can be reapplied elsewhere. The algorithm to determine the flood zones is an unsupervised change detection approach. This methodology is comparing before and after flood Sentinel 1 SAR satellite images by means of Google Earth Engine. An intersect operation between the flood zones and the Building Footprint determines the affected buildings.

The results vary highly depending on the size and spatial distribution of the flood zones. Factors like the selected algorithm, acquisition dates of satellite images, land surface of the study site and the parameter configuration for the algorithm have a high influence on the established flood zones. A part from these limitations, the results indicate that in a first damage evaluation the Building Footprint can be an added value to assess the affected buildings. For the selected flood event, the indicator total area covered by the affected buildings has turned out to be more suitable than the indicator number of affected buildings. The Building Footprint slightly overestimated the total area covered by the affected buildings compared to the OSM building dataset. This overestimation is small and can be explained, hence the indicator can be considered valid for the selected flood event. On the other hand, the results for the indicator number of affected buildings from the Building Footprint are not persuasive. The verification dataset OSM has significantly higher

numbers for the affected buildings. This is caused by the fact that the Building Footprint merges small buildings within densely populated urban areas to a single building.

The biggest potential for the established workflow and the Building Footprint data is most likely within rural areas. This is due to the fact that the flood detection algorithm is more susceptible to errors in urban areas and the data accuracy of the Building Footprint is lower within densely populated areas.

1 Introduction

Worldwide floods are the most common natural disaster and result in immense economic loss (Ferreira et al., 2011). The exposure of poor people to floods is disproportionally higher, especially in urban areas on the African continent (Winsemius et al., 2018). In the future the frequency and intensity of floods are expected to amplify due to climate change (IPCC, 2018, 2021). According to predictions by Hallegate et al. (2013) in 2050, solely in coastal cities, the annual losses due to flooding will reach a trillion US dollars.

The impact is also compounded by the fact that, urban growth in the global south is often taking place in areas that are prone to floods, because of limited housing space (Braun & Aßheuer, 2011). This trend is particularly worrying in Africa, as the urban population is growing rapidly. The annual growth rates are above 3 percent and it is predicted that by 2030 the two African cities, Luanda (Angola) and Dar es Salaam (Tanzania), will reach megacity status (United Nations, 2019). The Human Settlements Programme of the United Nations (2014) is estimating that Dar es Salaam has to accommodate more than 200’000 new urban dwellers per year.

Those factors indicate that the situation in the upcoming years may become worse and it is therefore crucial for fast growing cities in the south to adapt to this circumstance. Better planning and monitoring to adapt to this fast-changing environment would be crucial in helping cities to adapt and accommodate population and environmental pressures. Local decision makers and humanitarian aid agencies depend on accurate information in regards to flood events as fast as possible. Besides those directly tangled actors, institutions like insurance companies have an interest in knowing the expected impact.

Traditional methods that are used to quantify losses like the “The Damage, Loss and Needs Assessment” are often based on surveys, interviews and expert visits on site (Jovel & Mudahar, 2010). As a result of rapid technological changes and the availability of remote data, technical solutions become valued alternatives and have been used for damage quantification (Barnes et al., 2007; Fernandez-Galarreta et al., 2015). A current trend to assess damage in urban areas is to use high resolution remote sensing data either from satellites or UAV (Jiménez-Jiménez et al., 2020; Kakooei & Baleghi, 2017). Unfortunately, the needed high-resolution data for those assessments are often not available for developing countries, either due to high cost or missing equipment. Rahman & Di (2017) concluded, while providing an overview on remote sensing methods for flood mapping, further research is needed in low-cost and effective flood management strategy for developing countries.

As a result of the decision by the European Space Agency (ESA) Copernicus program to make their data from the Sentinel satellites freely available, new opportunities for low-cost solution have opened up. The Sentinel 1 satellites regularly provide SAR images from around

the globe. SAR based flood mapping is currently the most effective option to detect flood zones through remote sensing (Schumann & Moller, 2015). This is due to the fact that the microwave signals are sensitive to water and independent from the weather conditions during image acquisition (Notti et al., 2018). One of the most rapid and most common SAR based approaches to detect inundation zones is thresholding (Zhang et al., 2020). For example, the United Nations Office for Outer Space Affairs (UN-SPIDER) developed different “recommended practices” to detect and analyze hazards. One of these workflows is an unsupervised change detection threshold approach for flood mapping with Google Earth Engine (GEE). GEE is a web platform that provides cloud computing and an extensive data catalog (Gorelick et al., 2017). In recent years studies like Tiwari et al. (2020) and DeVries et al. (2020) showed that cloud computing platforms have become more popular within flood mapping, as they offer easy data access and fast computing of large data amounts.

The humanitarian sector has started to invest resources in mapping projects to enhance the spatial cover of relevant data. Accurate and up-to-date digitized building structures can be used as proxy information to evaluate the number of affected people and property damage. They have recognized the potential of community mapping and tools like Open Street Map (OSM) as a basis for data in disaster risk reduction (Scholz et al., 2018; Herfort et al., 2021). For example, to increase flood resilience in Dar es Salaam the community mapping project, “Ramani Huria”, trained volunteers to create highly accurate datasets of buildings and drainages (World Bank, 2016). The data was then used to create an “Atlas of Flood Resilience” in Dar es Salaam. One of the disadvantages of such projects is the limited range. Overall the inequality in data accuracy and reliability within OSM is problematic (Barron et al., 2014; Herfort et al., 2021).

In recent years automatic extraction of building footprints from satellite and aerial images with deep learning methods has made significant progress (Shu et al., 2018; Schuegraf & Bittner, 2019; Zhao et al., 2021). A main performance increase in image recognition is based on the invention of residual networks, which allow to scale deep neural networks (He et al., 2016). Compared to manual digitalization it is less time consuming, cheaper and the inequality in data accuracy is smaller. In 2018 Microsoft published a US building footprint with 125 Mio. automatic extracted buildings from Bing Maps (Microsoft, 2018). A year later, they published a building footprint for Uganda and Tanzania (Microsoft, 2019). In both cases object classification was done by deep learning methods (Tan & Le, 2020). The US building footprint has already been used for several data analyses (Heris et al., 2020; Jiang, 2019).

In this thesis, the intention is to create a method to assess the number of affected buildings during a flood event using the Building Footprint from Microsoft as input data. The study will focus on a flood event in Dar es Salaam. Verification is done by comparing the result with the building layer from OSM for a specific flood event. A comparison allows to answer the

question, can automatic extracted building footprints be used to evaluate the damage of flood events in different areas. If there is no significant difference in the outcomes, the method could be applied for other areas with less accurate OSM data availability.

In this research, using DSM, TZ as an example, the open data Building Footprint is used to assess the number of affected buildings due to a flood event.

In order to achieve this aim, the following objectives have been defined:

1. Create a workflow to estimate the affected buildings in the study site.
2. Verify the result by comparing it with another data source.
3. Analyze the potential and limits of the methodology and the Building Footprint in damage assessment of flood events.

2 Methods

The following chapter gives an overview about the selected study site and the selected flood event in section 2.1. A flood event was required to establish and verify the developed workflow. The created workflow to estimate the affected buildings is described in 2.2. The flood zone determination approach is explained in 2.3. The last section 2.4 describes the two datasets Building Footprint and OSM.

2.1 Study Site and Flood Event

For the following thesis the selected study site focuses on Dar es Salaam, Tanzania. It is centered on the districts of Ilala, Kinondoni and sections of Temeke, because these areas are the most heavily populated parts of Dar es Salaam. The selection is based on the population data provided by the Humanitarian Data Exchange Tool by the United Nations Office for the Coordination of Humanitarian Affairs OCHA (OCHA, 2016). According to the metadata, the data was originally collected by the Tanzania National Bureau of Statistics in 2012. The ward boundaries are based on the information from the Tanzania National Bureau of Statistics (NBS Tanzania, 2012). Figure 1 highlights the selected study site and the ward boundaries.

Flood events were identified by searching in past newspaper articles, which mention heavy rainfalls and floods for the region Dar es Salaam. According to local newspapers heavy rainfalls, starting on the 12 October 2020 and peaking on the 13 October 2020, caused extensive flooding in Dar es Salaam (Daily News, 2020; TheCitizen, 2020). The Copernicus Open Access Hub indicates that Sentinel 1 took an image on the afternoon of the 13th of October 2020 (ESA, 2020). The fact that a satellite image is available on the day of the flood peak makes it a good example to examine. Consequently, this flood event was chosen as an example to establish a workflow and verify the results. In order to determine the inundation zones, the methodology also needs a reference image before the flood (details see section 2.3). This image should be taken decidedly before a flood event starts. As a consequence, the time period for the before flood image was set on the first half of September 2020. Long term mean monthly rainfall data for Dar es Salaam shows that September is typically part of the dry season (Kabanda, 2018). This is confirmed by the aggregated rainfall data for September 2020 from the Agrometeorological Database of Tanzania. The records indicate less than 25 mm rainfall over the whole month (TMA, 2020).

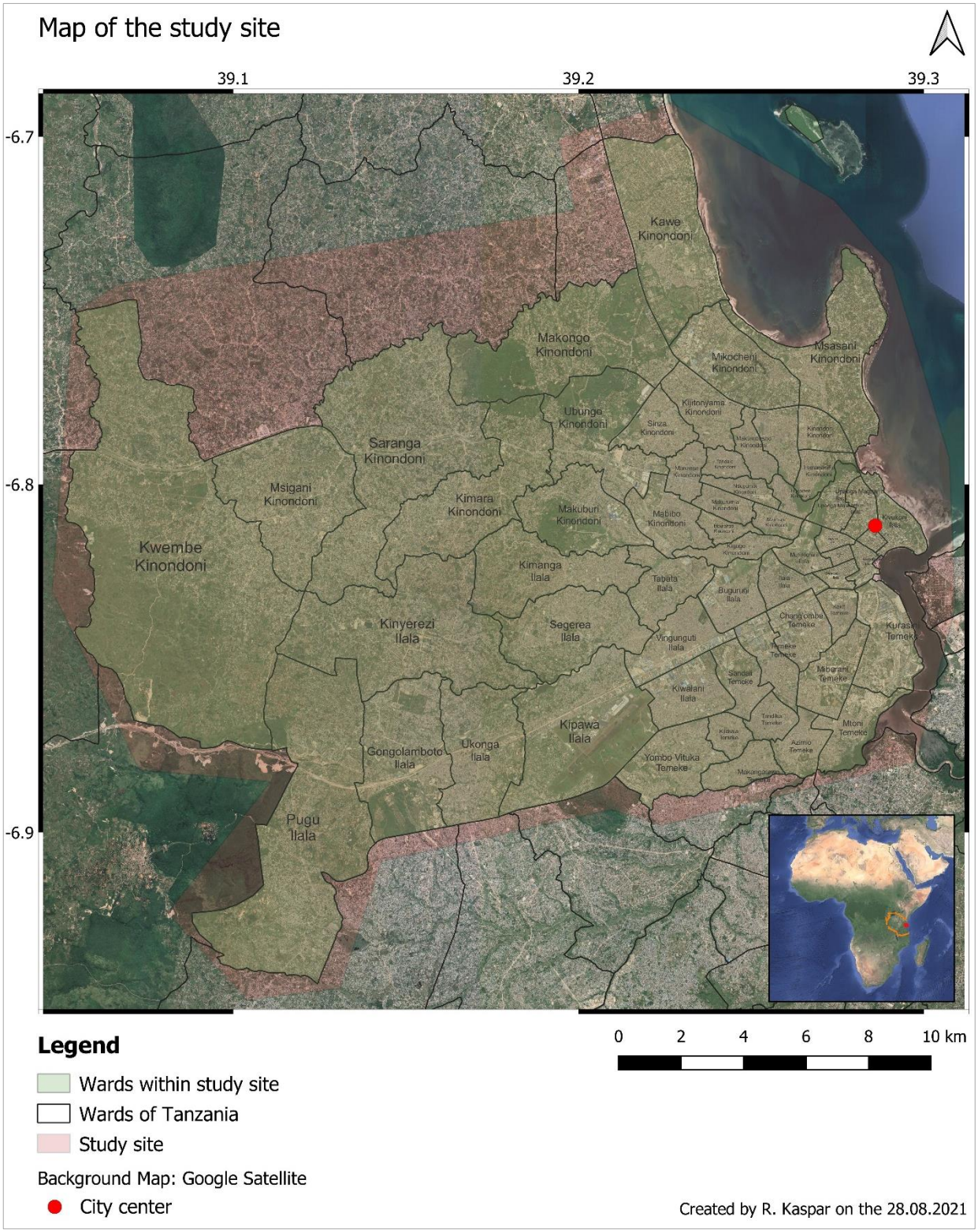

Figure 1: Map of the selected study site

2.2 Workflow

The aim of the workflow was to create an automatic procedure to calculate the two indicators number of affected buildings and total area covered by the affected buildings for the Building Footprint. In addition, the indicators are calculated for OSM data to help verify the quality of the result from the Building Footprint. A flood map is created to provide a visual result for the spatial distribution of the flooded areas and the affected buildings. The freeware tool QGIS was used to implement the workflow, details about the system set up can be found in the appendix in section 7.6.

The overall process of the methodology is summarized in Figure 2. The procedure is managed by a Python console script “Flood Map/Indicator”. The script calls step by step other processing scripts and QGIS algorithms. Some actions depend on intermediate results of earlier tasks, hence the sequence is critical. The order is pointed out by numbers in Figure 2. The modular structure of the procedure has the advantage of making it easy to replace each module. For example, if an alternative option to determine the flood zone is available, only the processing script “Create flood map” has to be replaced.

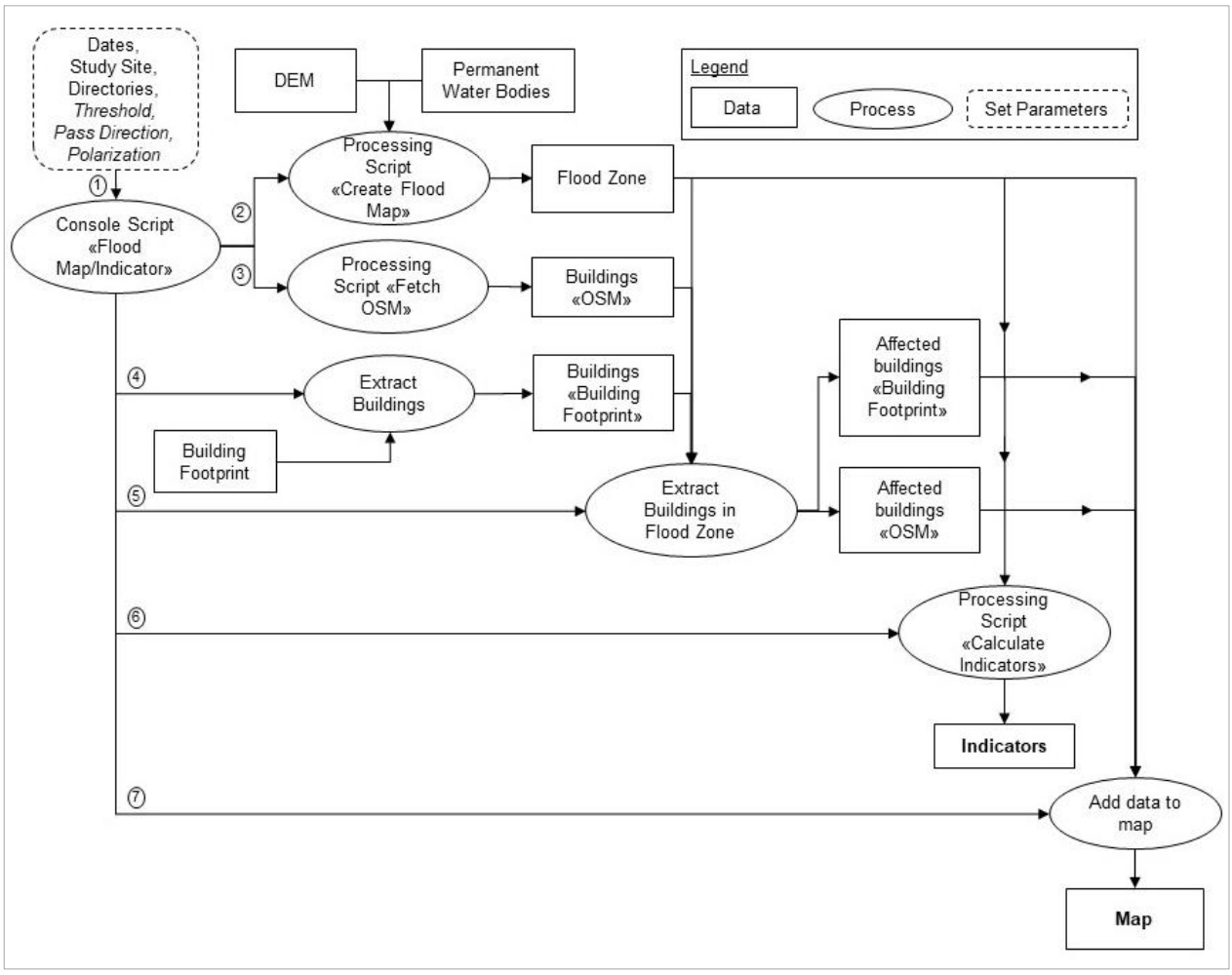

Figure 2: Overview of the procedure Calculation of indicators and map; order is indicated by numbers

The following key parameters need to be defined in the console script “Flood Map/Indicator” before running it:

- Time period before flood to select image (Dates)
- Time period to select flood image (Dates)
- Study Site (Path to local shapefile)
- Local Directories (Path to Building Footprint & Output Files)
- Threshold (Number)
- Pass Direction (Ascending or Descending)
- Polarization (VH, VV)

An example for the configuration of the script is described in detail in the appendix section 7.1.

The first algorithm called in the console script *Flood Map/Indicator* is the processing script *Create Flood Map,* see step number 2 in Figure 2. The script is attached in the appendix section 7.3. The result of the script is an inundation map based on the change detection approach (details see section 2.3). The script is using Google Earth Engine (GEE) to compare SAR satellite pre and post flood images. The initially defined parameters from the console script *Flood Map/Indicator* are handed over to the processing script and a request is sent to GEE. The images accessed in the Earth Engine Data Catalog are C-Band Sentinel-1 SAR GRD (Google Developers, 2020). These images are pre-processed with the Sentinel-1 Toolbox (Filipponi, 2019) and therefore, the only additional image processing step done was speckle reduction. Each pixel from the pre and post image is compared and classified according to the selected threshold in flood area or non-flood area.

Furthermore, elimination of false positive values is done by comparing the flood zone with additional datasets. Firstly, the flood zone are compared with a dataset of water bodies that occur for more than 10 months per year and all flood zone pixels intersecting the waterbodies are removed (Google Developers, 2021; Pekel et al., 2016). Next, all inundation zones that are placed in areas with over 5% degrees are removed. Identification and masking out these areas is done by comparing the flood zones with slope information based on the HydroSHEDS digital elevation model (Google Developers, 2008; Lehner et al., 2008). Finally, flood polygons that are smaller than 10 pixels are removed.

The flood zones are saved in the local directory defined in the console script *Flood Map/Indicator* as a TIF file. Then the TIF is converted into a SHP using the algorithm *gdal:polygonize* and invalid geometries are fixed with the algorithm *fixgeometries.* Finally,

flood polygons outside of the study site are eliminated by using the algorithm *clip* and the result saved as *floodedarea*.*SHP* in the local directory.

To calculate the affected buildings for the verification process, the next main step in the console script *Flood Map/Indicator* is fetching the OSM buildings within the study site, see step number 3 in Figure 2. This is done through the processing script *Fetch OSM*. The script is based on a model designed with the QGIS model designer. A query was created to download OSM polygon buildings in the study site (see section 7.4.1). The query uses the data mining tool *Overpass Turbo API* to access OSM data (Raifer, 2021). The result is saved in the local directory.

In order to calculate the affected buildings from the Building Footprint, the buildings for the study site are extracted, see step number 4 in Figure 2. The console script *Flood Map/Indicator* initiates a clip process to extract the building polygons from the Building Footprint and saves a copy of the data in the local directory.

The process is continued by extracting the buildings intersecting the flood zones, see step number 5 in Figure 2. Hence, the intermediate results, flood zone and the buildings from the Building Footprint, are used as input for the algorithm *extractbylocation* to define the affected buildings. The procedure is repeated with OSM as input data to extract the affected buildings for the OSM dataset. Both results are saved in the local directory.

The calculation of the indicator values is completed in the processing script *Calculate Indicators,* see step number 6 in Figure 2*.* The purpose of this processing script is to calculate the two indicators number of affected buildings and total area covered by these buildings*.* For each feature (building) in the dataset a field calculation is performed to determine the area of the polygon. Finally, the areas of the polygons are summed up and saved locally in the csv file *indicatorvalues*. Furthermore, the number of polygons is saved as well. This process is performed for the datasets *Affected buildings "Building Footprint"* and *Affected buildings "OSM".* In addition, in order to receive some useful auxiliary data about the study site the processing script is called for the datasets: *Flood Zone, Buildings "OSM",* and *Buildings "Building Footprint".* The results are all saved in the same csv file.

The final main step of the console script *Flood Map/Indicator* is adding the data, see step number 7 in Figure 2. *Study Site, Flood Zones, Affected buildings Building Footprint, Affected buildings OSM* are imported to the map. The data is placed in a group called Flood Event and appropriately formatted (names, colours, transparency).

In order to find an optimal threshold and polarization (see section 2.3 for explanation on selection of threshold and polarization), six different scenarios were calculated. The predefined threshold in the UN-SPIDER (2020) methodology is 1.25, hence this was the initial value for both polarizations. The values were reduced in steps of 0.05 until 1.10 for the

polarization VH and until 1.20 for the polarization VV. Table 1 gives an overview about the input parameters.

Table 1: Input parameters for console script Flood Map/Indicator

Before Flood Start Date	01/09/2020				
Before Flood End Date	20/09/2020				
After Flood Start Date	10/10/2020				
After Flood End Date	17/10/2020				
Pass Direction	ASCENDING				
Polarization / Threshold	VH	1.25	1.2	1.15	1.1
	VV	1.25	1.2		

2.3 Flood Zone determination

An unsupervised change detection and threshold approach was used to calculate the flood zones within the study site. This approach was modified from the recommended practices by the United Nations Office for Outer Space Affairs (UN-SPIDER, 2020). The method compares SAR satellite images before and after the flood. SAR satellites actively send electromagnetic pulses and receive the echoes of the backscatter (Moreira et al., 2013). The backscatter depends on the surface structure and accordingly on how smooth or rough the surface is, the signal appears bright or dark on the image (Lillesand et al., 2008). Therefore, the main element to detect flood areas while using SAR data is the shift of the surface roughness.

The energy transmitted from SAR sensors can be classified in different microwave frequencies. For change detection and monitoring of areas with low to moderate vegetation the C- band with a frequency of 4-8 GHz is the most suitable (Meyer, 2019).

Revisit times depend on the study site and the selected satellite. The satellite selected for the study site is Sentinel-1 SAR. In the case of the relevant study site (see Figure 1), only an ascending pass direction is covering the area completely and hence a complete image of the study site is only available every 12 days. In addition, a change detection approach depends on satellite images from the same viewing angle, when using satellite images from different pass direction, the comparison would lead to false positive signals (Lillesand et al., 2008).

Sentinel-1 SAR has four different modes. Over land the Interferometric Wide swath (IW) is the main acquisition mode and is suitable for the purpose of flood mapping (Geudtner et al., 2014).

The selected polarization of the signal is an important factor and has a high influence on the determined inundation zone as different polarizations do not interact equally with objects on

the surface. This is due to the fact that on different polarimetric channels the signal for scattering types like rough surface scatterers, double-bounce scatterers, etc. reacts differently (Meyer, 2019). According to UN-SPIDER (2020) the proposed polarization for flood mapping is vertical transmitted and horizontal received (VH) as it is more sensitive to land surface changes. On the other hand, Tiwari et al. (2020) preferred vertical transmitted and vertical received (VV) over VH polarization. Sentinel-1 SAR is provided in single co-polarization VV and dual-band cross-polarization VH (Google Developers, 2020). The focus of the research was the VH polarization, nevertheless two scenarios with the VV polarization were included for comparison (see Table 1).

A key parameter in change detection approach with a threshold, is choosing an appropriate threshold value. Each pixel from the preflood image is compared with the post flood image and then classified as under or above the threshold (Meyer, 2019). The methodology of UN-SPIDER (2020) follows a trial and error approach to define the threshold value. To avoid under or over classification of inundation zones, several scenarios have been calculated (see Table 1).

All SAR images are affected by speckle, which occur due to several individual scatters in a single pixel (Lillesand et al., 2008). These speckles appear as random patterns of bright and dark pixels in the image. This so called “Salt and Pepper” effect, can be reduced by applying a filter on the image. The methodology is applying a simple local mean filter to reduce the speckles on the pre and post flood images.

2.4 Datasets OSM and Building Footprint

OSM is a free map project formed on a crowdsourcing approach. Everybody can add and edit content like buildings, waterways, etc. In recent years there is a growing interested from researches to use OSM data, as the quantity of free available data builds a huge opportunity (Sehra et al., 2017).

The free available Building Footprint used in the procedure was created by Microsoft’s AI for Humanitarian Action program (Microsoft, 2019). The dataset includes almost 18 million building polygons for Tanzania and Uganda, whereas more than 11 million lay in Tanzania. The data is published as GeoJSON format. The building polygons were established in a two-step process (Microsoft, 2019). Firstly, a semantic segmentation is done by using a Convolutional Neural Networks (CNN) to extract building pixels from satellite images. The CNN used is called EfficientNet B3 (Tan & Le, 2020). Secondly, polygons are built based on the established pixel blobs. The source of the satellite images used in the process is Bing Maps and therefore it is not possible to establish the exact date of the acquisition of individual images (Microsoft, 2019). According to Microsoft (2019) the quality of the Building Footprint is mostly at least as good as hand digitized buildings in OSM. They computed a precision value

of 94.5%, a recall factor of 61.8% and an Intersection over Union value (IoU) of 68% (Microsoft, 2019). However, precision, recall factor and IoU are lower in dense urban areas compared to rural areas (Microsoft, 2019).

A copy of the GeoJSON file needs to be stored as a SHP file locally on the computer (see section 7.1). This has to be done to raise the efficiency of the procedure as it increases the calculation process significantly.

3 Results

This chapter present the results for the indicators and the maps generated through the methodology. The maps allow a visual interpretation and hence give additional information about the spatial distribution of the flood zones and affected buildings. All areas calculated in the tables are listed in hectares.

Table 2 sums up general information for the study area including the size and the number of buildings for the datasets *OSM* and *Building Footprint*. Furthermore, the sum of the area per dataset is given. Table 3 shows the area of the flood zone and the number of affected buildings for scenarios with VH polarization on both datasets. The results are split up by thresholds, 1.25, 1.20, 1.15 and 1.10. This table also lists the total area covered by the affected buildings. Table 4 contains the results for the scenarios with polarization VV and threshold 1.25 and 1.20. Again, the number of affected buildings and the total area of affected buildings are displayed.

Table 3 and Table 4 illustrate that a decreasing threshold, increases the size of the flood zone and therefore, the number of affected buildings. This is visually confirmed in Figure 8, which points out the effect of the parameter threshold on the flood zone polygons. Low threshold values like 1.15 and especially 1.10 generate additional small inundation zones.

Depending on selected polarization and threshold, the area classified as being flooded ranges between 0.26% and 0.74% of the study site. In absolute numbers total flood zone sizes range from 140 hectares to 411 hectares (see Table 3 & Table 4), of which the scenarios with VV polarization result in higher numbers than the VH polarization. For example, the number of flood zone polygons with a threshold 1.2 is more than 9 times higher for VV, whereas the summed area is only around 2.2 times higher. Hence, the VV polarization creates more small and medium size flood zone polygons, evenly distributed through the study site. The different spatial distribution of the inundation zones over the study site are well illustrated when comparing Figure 3 and Figure 4. The VV polarization in Figure 4 shows conspicuously additional small and medium size flood zones distributed evenly all over the study site. As seen in Figure 3 and Figure 6 the main flood zones for the VH polarization are in the city center and along major rivers. An enlarged comparison is presented in Figure 5, which emphasizes the effect of the polarization on the number of flood polygons detected.

Table 2 shows that the number of buildings in the *OSM Building Layer* is more than three times higher than the number of buildings in the *Building Footprint Layer*. On the other hand, the area covered by the *OSM Building Layer* is only around 50% higher than the area covered by the *Building Footprint Layer*. This indicates that the individual polygons of the *Building Footprint Layer* are bigger. This is confirmed in Table 3 and Table 4 by looking at the

numbers for affected buildings and the area that those buildings cover for each of these datasets. The number of affected buildings is for all scenarios significantly higher for the *Affected buildings OSM Layer.* On the other hand, the area covered by the *Affected buildings Building Footprint Layer* is always slightly higher than the total area covered by the *Affected buildings OSM Layer*.

Table 5 shows ratio calculations for the values in Table 3 and Table 4. As mentioned above, the number of affected buildings is significantly higher for the *OSM* dataset. The ratio for the *Affected buildings OSM Layer* to *Affected buildings Building Footprint Layer* outlines this in a single number. Independent from polarization and threshold, the ratios of the number of affected buildings range between 200 and 300 percent. On the contrary, the ratio *Area of Affected buildings OSM Layer* to *Area of Affected buildings Building Footprint Layer* only ranges from 88.8% to 99.4%. This means even though the OSM dataset produces higher numbers of affected buildings, the total areas covered by the Building Footprint are slightly higher. This seeming contradiction can be explained by the circumstances that the study site and the established flood zones are mainly in areas where housing units are very densely packed. The *Building Footprint* dataset often combines several small housing units to one big polygon and therefore, the number of buildings is lower compared to the *Buildings OSM Layer.* This can be visually seen in Figure 7, an enlarged view from Figure 3. It shows clearly the difference in level of detail between the affected buildings from the OSM dataset and the Building Footprint. The difference between the total areas of the two datasets are for all scenarios less than 12%. The VV polarization shows for both thresholds an almost identical size for the area with differences of less than 1%. This is pointed out by the ratio *Area of Affected buildings OSM Layer: Area of Affected buildings Building Footprint Layer* in Table 5.

Table 2: Overview general numbers of the study site

Dataset Name		**Number of polygons**	**Total area of polygons in hectares**
Study Site Area		1	54'411.46
OSM Building Layer		749'420	7'674.27
Building Footprint Layer		236'939	4'933.98

Table 3: Overview Indicator values (absolute numbers) for VH polarization

Dataset Name		Number of polygons	Total area of polygons in hectares
Affected buildings OSM Layer	Threshold: 1.25	889	31.36
Affected buildings Building Footprint Layer		408	34.03
Flood Zone		141	140.27
Affected buildings OSM Layer	Threshold: 1.20	1397	42.01
Affected buildings Building Footprint Layer		567	46.23
Flood Zone		176	188.67
Affected buildings OSM Layer	Threshold: 1.15	2363	66.95
Affected buildings Building Footprint Layer		877	75.00
Flood Zone		269	255.01
Affected buildings OSM Layer	Threshold: 1.10	4453	103.80
Affected buildings Building Footprint Layer		1522	116.85
Flood Zone		555	369.93

Table 4: Overview Indicator values (absolute numbers) for VV polarization

Dataset Name		Number of polygons	Total area of polygons in hectares
Affected buildings OSM Layer	Threshold: 1.25	8162	164.38
Affected buildings Building Footprint Layer		2894	165.33
Flood Zone		1503	349.94
Affected buildings OSM Layer	Threshold: 1.20	9163	178.34
Affected buildings Building Footprint Layer		3229	179.43
Flood Zone		1629	411.58

Table 5: Overview of ratio for indicator values

	Polarization VH				Polarization VV	
	Threshold 1.25	Threshold 1.20	Threshold 1.15	Threshold 1.10	Threshold 1.25	Threshold 1.20
Ratio Affected buildings OSM Layer: Affected buildings Building Footprint Layer	218%	246%	269%	293%	282%	283%
Ratio Area of Affected buildings OSM Layer: Area of Affected buildings Building Footprint Layer	92.2%	90.9%	89.3%	88.8%	99.42%	99.39%
Ratio Flood Zone: Total Area	0.26%	0.35%	0.47%	0.68%	0.64%	0.76%

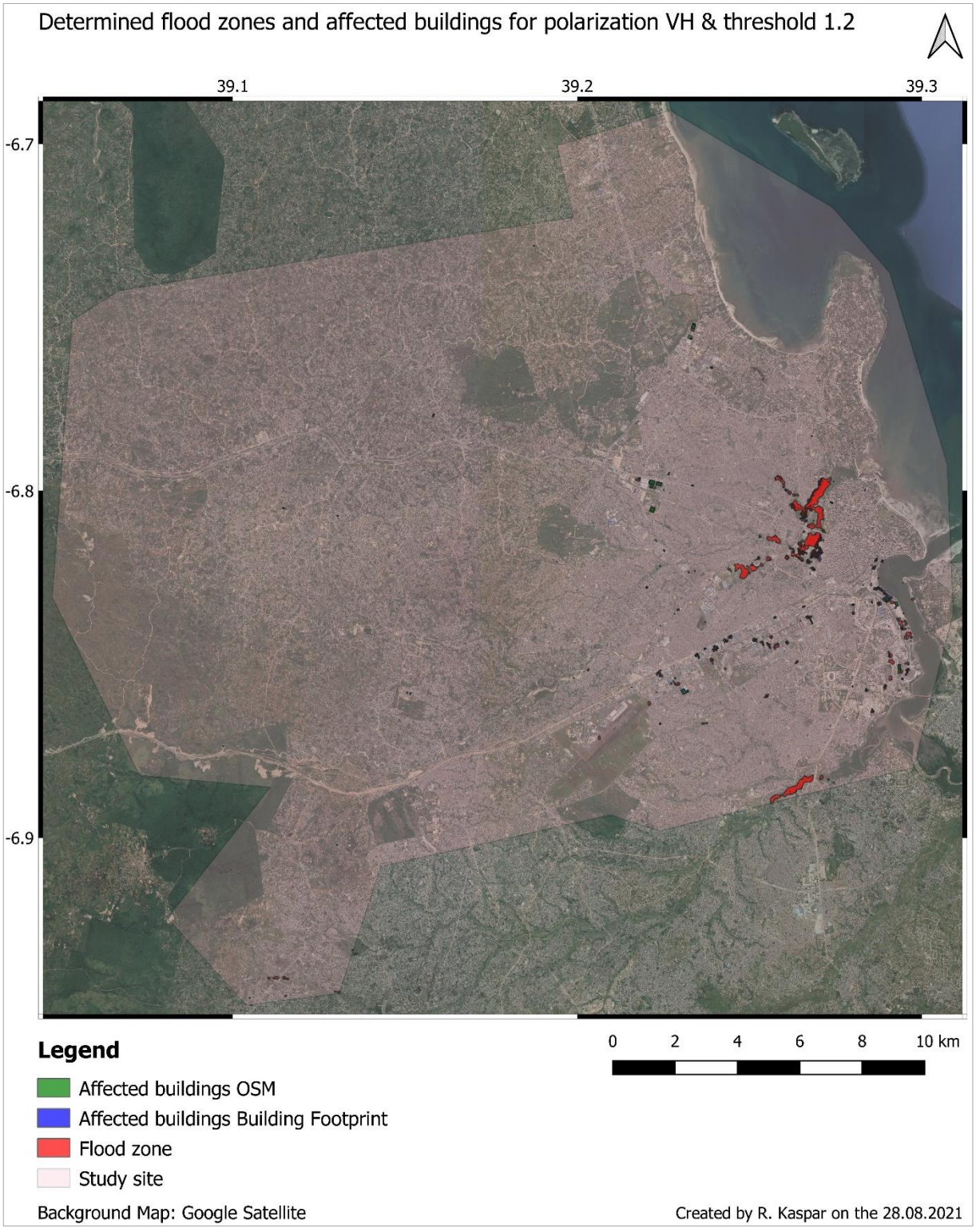

Figure 3: Map of study site, determined flood zones and affected buildings for OSM as well as Building Footprint; Background Google Maps; Threshold 1.2; Polarization VH

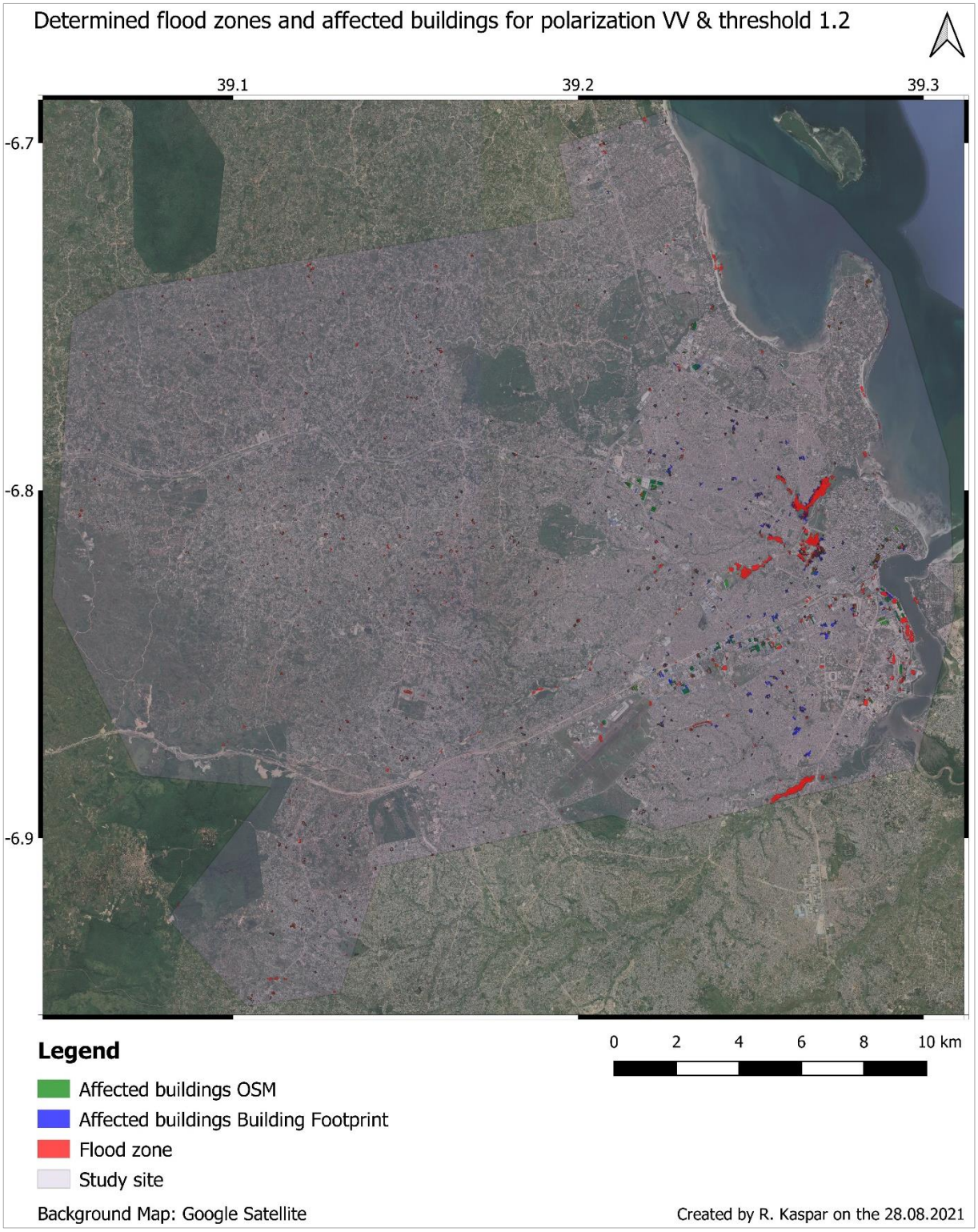

Figure 4: Map of study site, determined flood zones and affected buildings for OSM as well as Building Footprint; Background Google Maps; Threshold 1.2, Polarization: VV

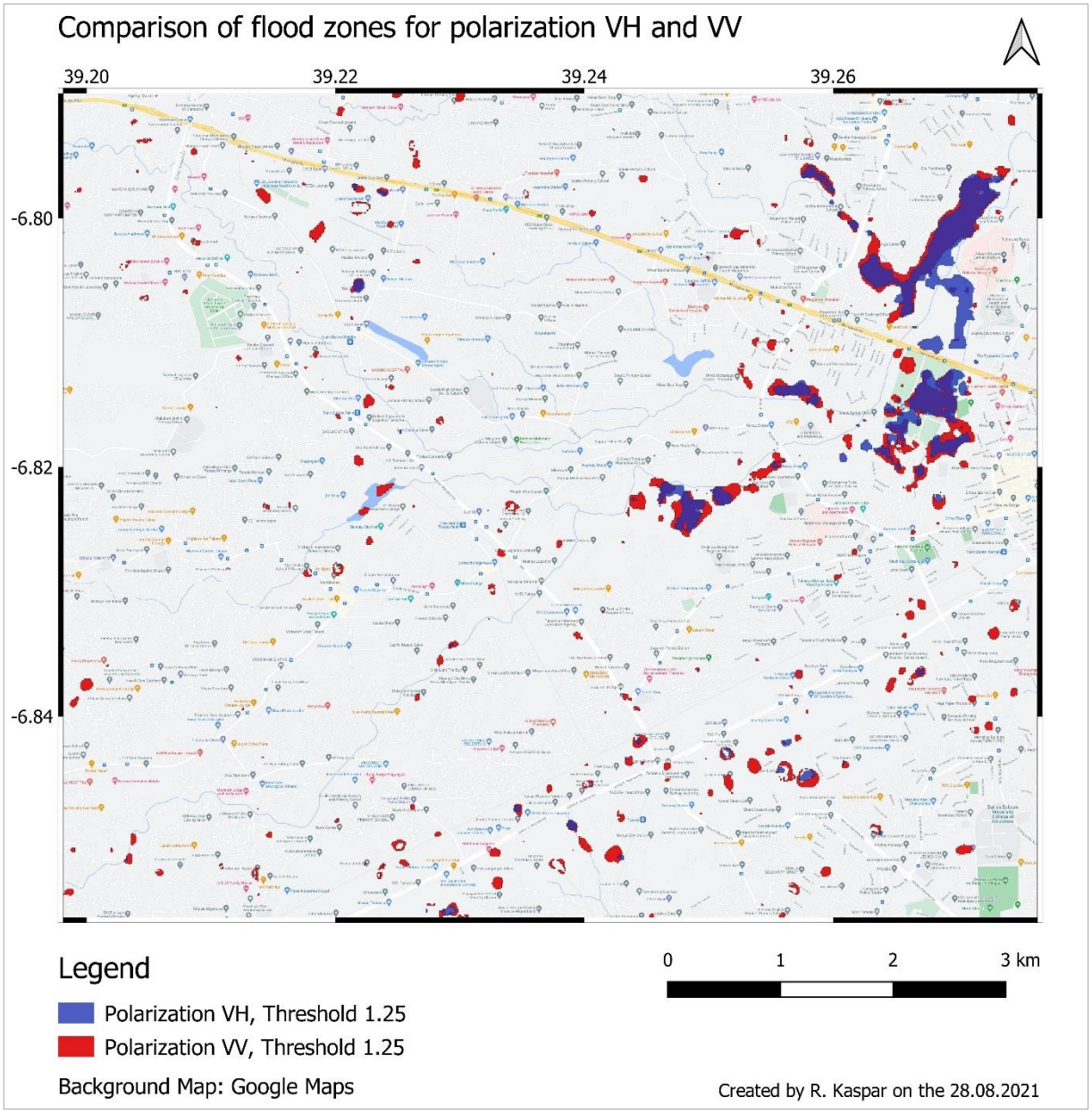

Figure 5: Comparison of flood zones for polarization VH and VV threshold 1.25

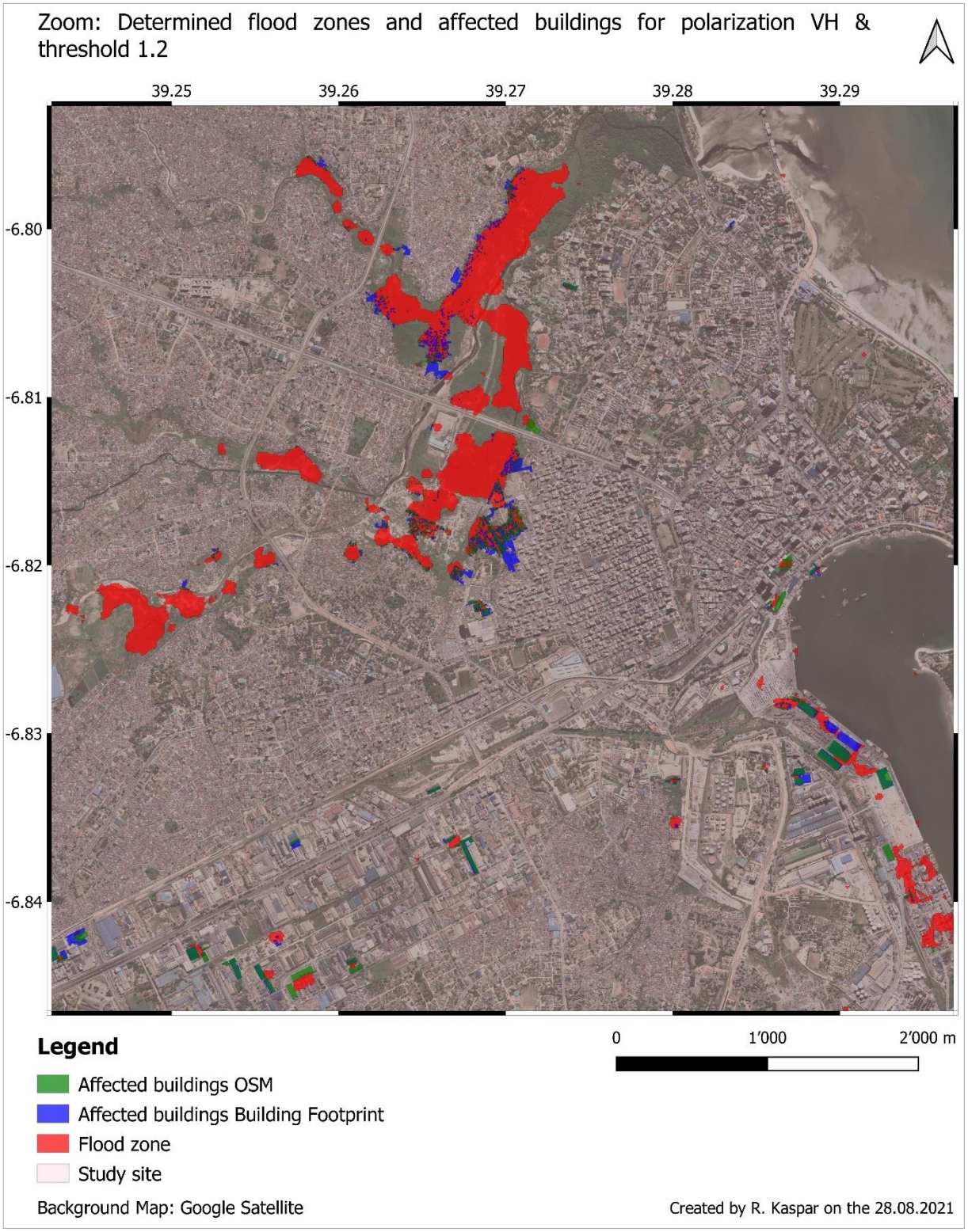

Figure 6: Zoom of main flood areas and affected buildings for OSM as well as Building Footprint; Background Google Maps; Threshold 1.2; Polarization VH

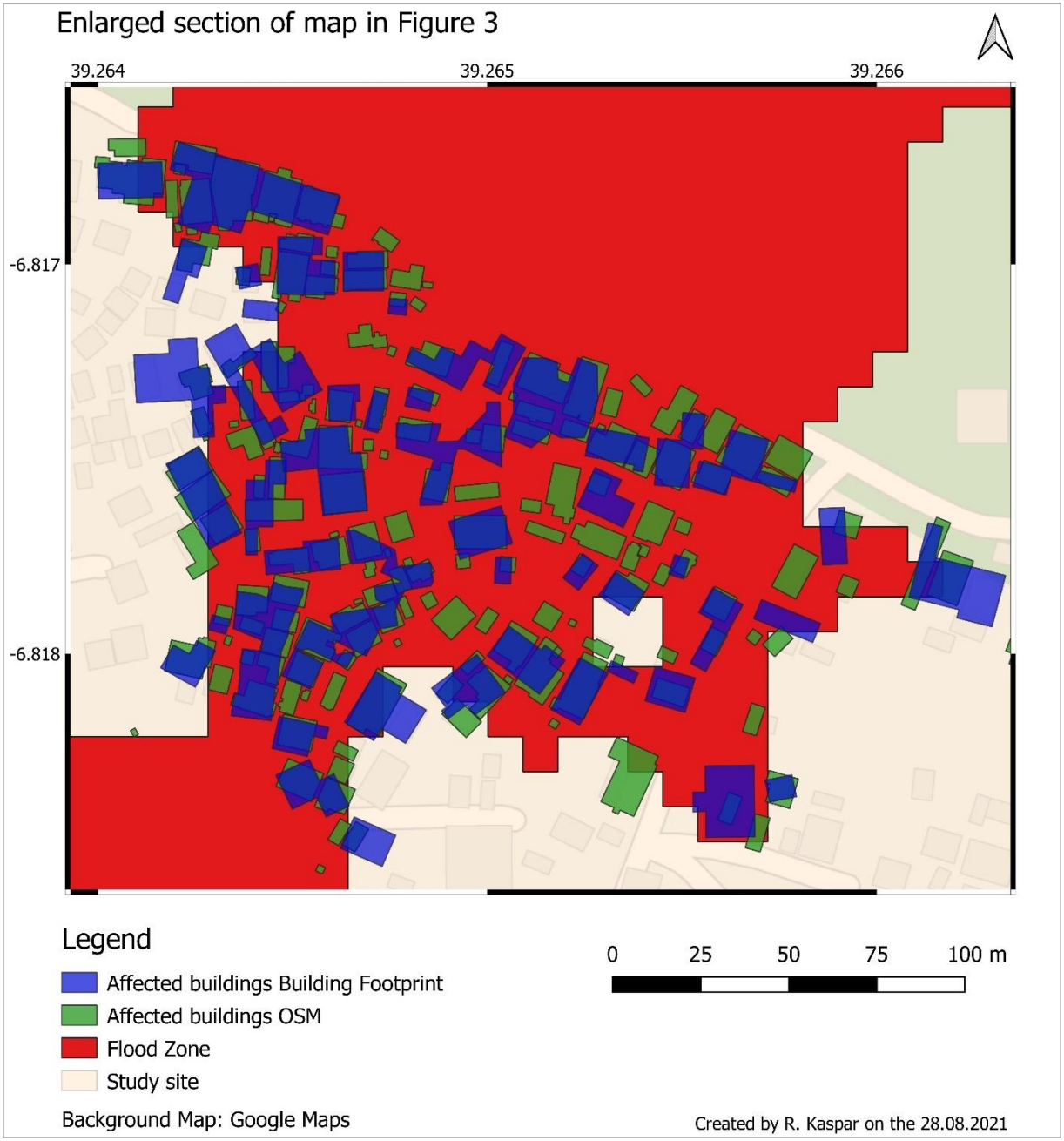

Figure 7: Enlarged section of map in Figure 3

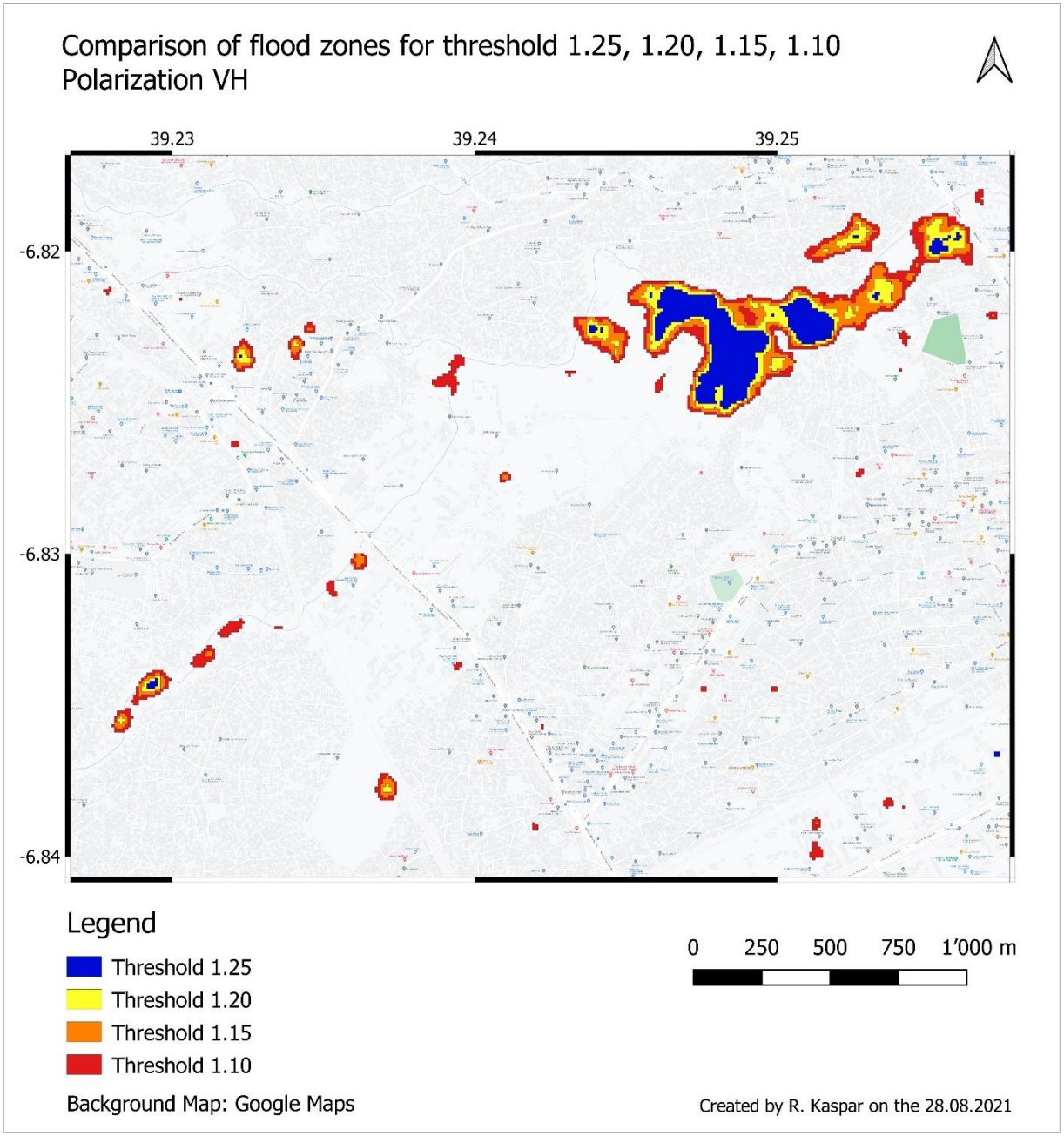

Figure 8: Comparison of flood zones for threshold 1.25, 1.20, 1.15, 1.10, Polarization: VH

4 Discussion

The discussion is divided into three sections. The first section of the discussion focuses on the algorithm to establish the flood extent. The second section looks closer at the quality of the indicators and the calculated results. The last section focuses on the question, whether the established workflow and the Building Footprint could be used to evaluate the damage of flood events in other areas.

4.1 Flood zone algorithm

As shown in Table 3 and Table 4 the areas of the established flood zones vary highly depending on the selected parameter. The focus of this study was not to establish a new algorithm to determine flood zones, but rather on the usability of a Building Footprint for a damage assessment. Nevertheless, the calculation for the numbers of affected buildings and the total area of affected buildings are based on the spatial distribution and the size of the flood zones. Therefore, it is important to look in this first section in detail at the process of the algorithm to determine the flood zones.

Sentinel 1 data was chosen in this thesis, as flood mapping is preferably done with active sensors. The key advantage of active sensors and therefore microwaves is that they manage to penetrate clouds, whereas passive sensors need a cloud free sky and this is rather seldom during flood events. In addition, a further advantage is active sensors do not rely on daylight. Schumann & Moller (2015) concluded, flood detection is only feasible with SAR imagery. On the other hand, Notti et al. (2018) pointed out that SAR images show only better results, when image acquisition is possible during the time of maximum inundation. When satellite images are only available a couple of days after maximum inundation, multispectral images showed better results for flood mapping (Notti et al., 2018). Unquestionably the pass time of the satellite is a key factor to get accurate results. Fortunately, the number of satellites with active sensors has increased over the last couple of years. In addition, further missions like the NASA-ISRO NISAR mission and the Biomass mission by ESA will soon provide additional free accessible SAR data.

Another significant factor is the technique selected to reduce speckle noise. The selection should be based on the requirements of the methodology and the characteristics of the input data. This is because, speckle filters with high noise removal tend to cause loss of image details (Rana & Suryanarayana, 2019). Instead of using a simple local mean filter as in the UN-SPIDER (2020) methodology applied in this thesis, alternative procedures like (enhanced) Lee filter or (enhanced) Frost filters could be applied. Recent studies in the area of flood mapping are often using a Lee filter (Agnihotri et al., 2019; Tiwari et al., 2020; Zhang et al., 2020), nevertheless median filters are applied as well (Anusha & Bharathi, 2019). Rana

and Suryanarayana (2019) evaluated different SAR filter techniques for flood mapping and concluded the best performance for despeckling and feature preservation is reached by the Lee filter.

Even the type of landscape in the study site can greatly affect which data and methodology is the most appropriate when analysing flood events. The backscatter effect reacts differently depending on the surface cover. Figure 9 displays the backscatter mechanism under non-flood and flood conditions for different surface covers. For urban areas the double bounce effect is dominant under non-flood and flood conditions, however the portion of the backscatter is usually higher under flooded conditions (see Figure 9, red square, size of return arrow). The challenges for urban flood mapping with SAR are diverse. Sealed surfaces like roads can be misclassified due to the low backscatter difference between the before and after image (Mason et al., 2014). SAR are side-looking instruments, buildings in urban areas cause radar shadowing and layover on the images and can lead to misclassifications (Giustarini et al., 2013). Layovers appear bright on the satellite images, hence they are always classified as non-flooded but could be flooded. Shadows are the opposite, they appear dark on the images same as water and therefore non-flooded areas can be classified as flooded. In 2019 while looking at the state-of-the-art algorithms using SAR images for mapping flood zones, Shen et al. (2019) concluded that there is no solution yet that is fully satisfactory for urban flood determination.

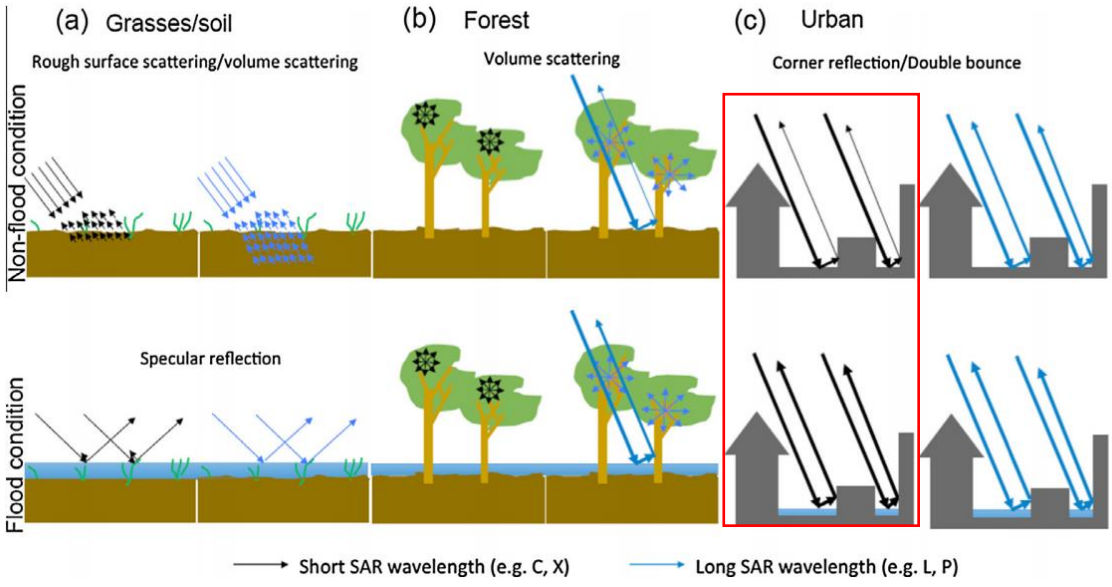

Figure 9: Backscatter from different surface covers (Schumann & Moller, 2015)

Furthermore, it is important to remember that, even for the chosen methodology the resulting flood zones vary depending on the parameter settings. The two parameters, polarization and threshold, have a high influence on the spatial distribution of the flood zones and the total area covered by the flood. The selection of these parameters is crucial. This is demonstrated

well by comparing the results for the different polarization and thresholds (see Figure 5 & Figure 8).

In the research literature there is disagreement in regards to which polarization is preferable for flood mapping. The proposed polarization by UN-SPIDER (2020) is VH, as it is more sensitive to land surface changes. On the other hand, Manavalan et al. (2017) found, for flooded areas within settlement, almost the same result for both polarization VH and VV. Other studies preferred VV polarization over VH, as it led to a higher accuracy in flood detection (Agnihotri et al., 2019; Tiwari et al., 2020). Due to this disagreement on which polarization works best, the scenarios calculated in this thesis, included VH and VV in order to ascertain which one is best suited for the selected flood event.

The UN-SPIDER (2020) methodology is following a manual selection of the threshold value. The selection is based on a heuristic approach. According to Malinowski et al. (2015) and Manjusree et al. (2012) manually deriving an optimal threshold requires information from training samples. Here, reference points from the field or optical satellite images are used to verify the results and optimize the threshold value. Unfortunately, training samples for the study site were not available, hence this thesis calculated several scenarios with different threshold and polarization for the flood zone determination (see Table 1). These scenarios are compared in order to determine if an optimal threshold and polarization can be found. Threshold selection started for both polarization from 1.25, which is the predefined threshold in the UN-SPIDER (2020) methodology. The value was reduced in steps of 0.05 until 1.10 for the polarization VH. For the polarization VV a threshold of 1.2 already led to a high number of small flood zone polygons, most likely false positive detections and therefore scenarios with lower threshold were avoided. The enlarged section comparing the flood zones for polarization VH and VV in Figure 5 shows clearly the higher number of small flood polygons for VV polarization. It cannot certainly be concluded that these small inundation areas for the VV polarization are false positive results, but these flood zones are suspicious. Threshold under 1.2 are most likely leading to false positive classifications of flood zones. A visual analysis of the map shows an increase of small flood zone polygons for threshold 1.15 and 1.10 (see Figure 8). The calculated indicators imply this as well, the number of flood zone polygons increases fast for lower thresholds. Reducing the threshold from 1.25 to 1.20 increases the number of polygons by only 25%, whereas the number increases up to 106% by altering the threshold from 1.15 to 1.10. A final conclusion on the best polarization and threshold is not possible, but thresholds under 1.2 lead most likely to false positive flood zone detection.

According to UN-SPIDER (2020) the change detection approach followed in the methodology leads to reliable results to determine flood zones after a major flood event. It is an unsupervised threshold algorithm and therefore one of the simpler approaches to establish a

flood extent. However, there are numerous other approaches that could be used and a one size fits all algorithm has not been found yet. Instead of using the UN-SPIDER (2020) methodology, an alternative algorithm to determine the flood zones could eliminate the step of threshold selection. Otsu (1979) presented an unsupervised method of automatic threshold selection from gray level histograms. It is still regularly used for flood mapping nowadays, for example in Cao et al. (2019) and Kordelas et al. (2018), as it is a simple algorithm. In recent studies the following approaches for flood mapping were applied and these methods reached reasonable results: DeVries et al. (2020) developed an algorithm to compare temporal SAR Z-scores with a time series of non-flood images to evaluate flood zones. Huang & Jin (2020) proposed a supervised change detection approach using SAR and optical images to map floods. Mason et al. (2021) established a change detection method to map floods in urban areas. Here, areas of increased backscatter, as a result of double bounce, are identified and further used to interpolate a flood level surface. The flood level surface is then compared with a digital surface model to determine the flood zone.

It can be concluded that the establishment of flood zones, especially in urban areas, is difficult and a variety of error sources exist in the detection of the inundation areas. Furthermore, the UN-SPIDER (2020) methodology to establish flood zones is particularly sensitive to threshold and polarization parameter selection. A conclusive determination of which scenario is closest to reality without reference points from the ground is problematic.

4.2 Meaningfulness of indicator

Independently from the algorithm to determine the flood zones, the quality of the calculated indicators from the Building Footprint can be assessed based on the calculated results. The different results from the scenarios offer the opportunity to compare the calculated results, as if they would be different flood events. The only parameter changing is the determined flood zone. Useful calculations for the verification step are the *Ratio Affected buildings OSM Layer* to *Affected buildings Building Footprint Layer* and *Ratio Area of Affected buildings OSM Layer* to *Area of Affected buildings Building Footprint Layer* in Table 5. The assumption is, the closer the numbers are to each other, the better is the result. Therefore, the ratios show the divergence between the calculated values for the two datasets. However, some facts need to be considered while analysing the results: The building polygons from the public available OSM data for Dar es Salaam have a high level of detail, due to the community mapping project “Ramani Huria” in Dar es Salaam (World Bank, 2016). As a consequence, the data is therefore a well base for the verification process within the selected study site. The quality of the OSM layer outside the region of Dar es Salaam is uncertain. Herfort et al. (2021) revealed significant OSM data inequalities across countries and within regions. The most critical point to keep in mind in the verification process, while comparing the OSM

buildings with the buildings from the Building Footprint, are the different acquisition dates. The created algorithm is requesting each time up-to-date OSM building polygons. The OSM data was fetched on the 8 of August 2021, while calculating the result in chapter 3. On the other hand, as mentioned in the methodology, the Building Footprint is based on Bing Maps. Bing Maps is a mosaic of satellite images and the exact date of acquisition is not known. The publication of the Building Footprint took place on the 16 of September 2019, this means all images used to establish the Building Footprint are older. Optimally the comparison would be done with building layers from the same time frame.

The first indicator calculated is the number of affected buildings. The number of affected buildings from the Building Footprint is underestimated for urban areas. As displayed in Table 3 and Table 4 the numbers for the affected buildings from the Building Footprint are in all scenarios significantly lower than the numbers from the OSM data. The OSM data shows twice to three times higher absolute numbers for the affected buildings. The results in Table 5 *Ratio Affected buildings OSM Layer* to *Affected buildings Building Footprint Layer* does point this out once more in a single number per scenario. This indicates that the data from the Building Footprint is not optimal for the affected building indicator. The OSM data source is a better option to define the exact number of buildings affected in urban areas. As mentioned in chapter 3, this difference is intensified by the fact that the major flood zones are in dense urban areas. In these areas, the Building Footprint incorrectly consolidates several small housing units into a single large building (see Figure 7). In this case the algorithm used to create the building polygons is not as precise. According to Shu et al. (2018), detection of small and dense buildings is a challenging task due to similarities between objects, low contrast with background and noise disturbance. Instead of creating several small buildings, the algorithm connects classified pixels next to each other into one big polygon (see Figure 10, left image). This could be heavily influenced by the spatial resolution of the original satellite image being too low. A pixel can only be classified as building or non-building and if the distance between two houses is less than the spatial resolution of the satellite image, the area between the houses cannot be classified as non-building. This leads to bigger building polygons within dense urban areas compared to the hand digitized OSM data. A higher resolution satellite image would lead to a better building layer and therefore increase the accuracy of the indicator *Affected buildings Building Footprint Layer.* The algorithm used to extract the Building Footprint is published on GitHub (Tan & Le, 2020). Therefore, it is theoretically possible to establish a better Building Footprint with a high-resolution and newer acquisitioned satellite image.

The second indicator is the total area of affected buildings from the Building Footprint Layer. The results are promising and the indicator is more suitable than the numbers of affected buildings. As shown in Table 3 and Table 4 the calculated areas for the affected buildings

from the Building Footprint are in all scenarios close to the numbers from the OSM data. In all calculated scenarios the difference between the Building Footprint and the OSM numbers is less than 12% (see Table 5). The calculated numbers for the total area of the Building Footprint are likely to be slightly overestimated. This can be explained through the following two factors:

- As explained in the last paragraph above, in urban areas with dense buildings the space between houses is often classified as buildings, this increases the area of the building polygons (see left image in Figure 10).
- According to the methodology all buildings that intersect with the flood zone are classified as affected buildings. The proportion that lies within the flood zone is irrelevant. Larger buildings, which only partly lie in the flood zone, increase the total area of the affected buildings. As established the Building Footprint tends to have larger polygons than the OSM layer and is therefore more affected by this effect (see right image in Figure 10).

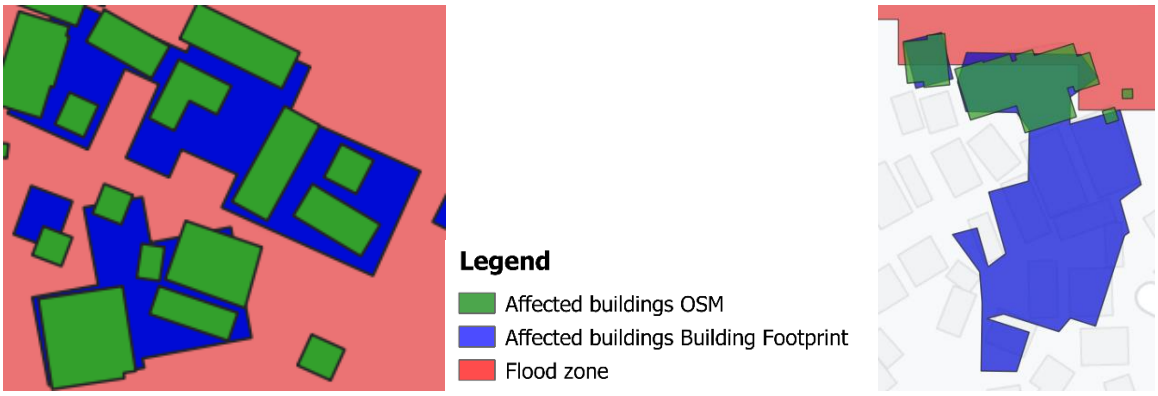

Figure 10: left image: Difference of the building perimeter between OSM and Building Footprint right image: Consequences of intersect effect for OSM and Building Footprint

According to Table 2 the total area covered by the buildings of the Building Footprint in the study site is 4933.98 hectares, while the total area covered by the OSM data is around 50% higher (7'674.27 hectares). Therefore, it could be concluded that the comparison of the total area of the affected buildings between the two datasets should result in a OSM value, which is around 50% higher. However, as mentioned above, the acquisition dates of the two datasets are different. There is almost a two-year time difference and meanwhile a vast building process took place within the study site (United Nations Human Settlements Programme, 2014). This explains, at least partly, the higher total area covered by the OSM data. The following analysis confirms this hypothesis. Comparing the total area covered by buildings, the two datasets show increasing differences for wards that reside further from the city centre. For example, the outer ward Kwembe, where there is still plenty of building space available, has for the OSM data 3.3 times more area covered (336%) than for the Building Footprint. This is different in city centre wards like Mchikichini, where there is less space for

new buildings available. There the area covered by the OSM data is only 15% higher than the area covered by the Building Footprint. Furthermore, an important element that must be considered is the spatial distribution of the flood zones. Large parts of the determined flood zones are around the city centre wards. Therefore, the outer wards with the higher variance of the building area coverage between the two datasets are less relevant for this flood event.

4.3 Transferability of the method

In this last section, the focus is on analysing to what extent the developed methodology and the Building Footprint could be used to evaluate the damage of flood events in other areas.

There is potential to use the methodology and the Building Footprint for assessing the level of flood damage in rural areas. The selected study site of Dar es Salaam is for the most part highly populated and buildings are packed densely. Here the OSM data has produced more accurate values for the indicator number of affected buildings. The values from the Building Footprint for total area of affected buildings are almost identical and hence more suitable as an indicator for urban areas. Nevertheless, the usage of the Building Footprint in urban areas relies on a regularly published up-to-date version, due to the fact that the covered area by buildings in large African cities can change quickly.

As mentioned earlier the OSM data has a high level of detail for the study site, but in rural areas this is not always the case (Herfort et al., 2021). Moreover, the accuracy of the data in the Building Footprint is better in rural areas (Microsoft, 2019). According to the metadata of the Building Footprint, the three quality parameter of the dataset, precision, recall factor and IoU are higher in rural areas, particularly the recall factor and IoU (Microsoft, 2019). This can be observed and confirmed, even in less populated areas of the study site. Figure 11 displays a comparison of the Building Footprint and OSM data for a less densely populated area within the study site. The number of recognized buildings is almost identical. The building perimeters established by the Building Footprint methodology are much more accurate compared to dense areas (compare Figure 7 & Figure 10) and consequently the area covered by the Building Footprint is more precise.

As mentioned above, several challenges in flood mapping with SAR occur mainly in urban areas (sealed surfaces, double bounce, shadows, etc.). Even according to UN-SPIDER (2020) the flood mapping methodology is limited, when it comes to detecting floods in urban areas. Mason et al. (2021) concluded that flood detection in rural areas with SAR images tend to work well. In all probability the flood zone determination process would lead to a better result in rural areas. Consequently, the correctness of the calculated indicators would increase as well.

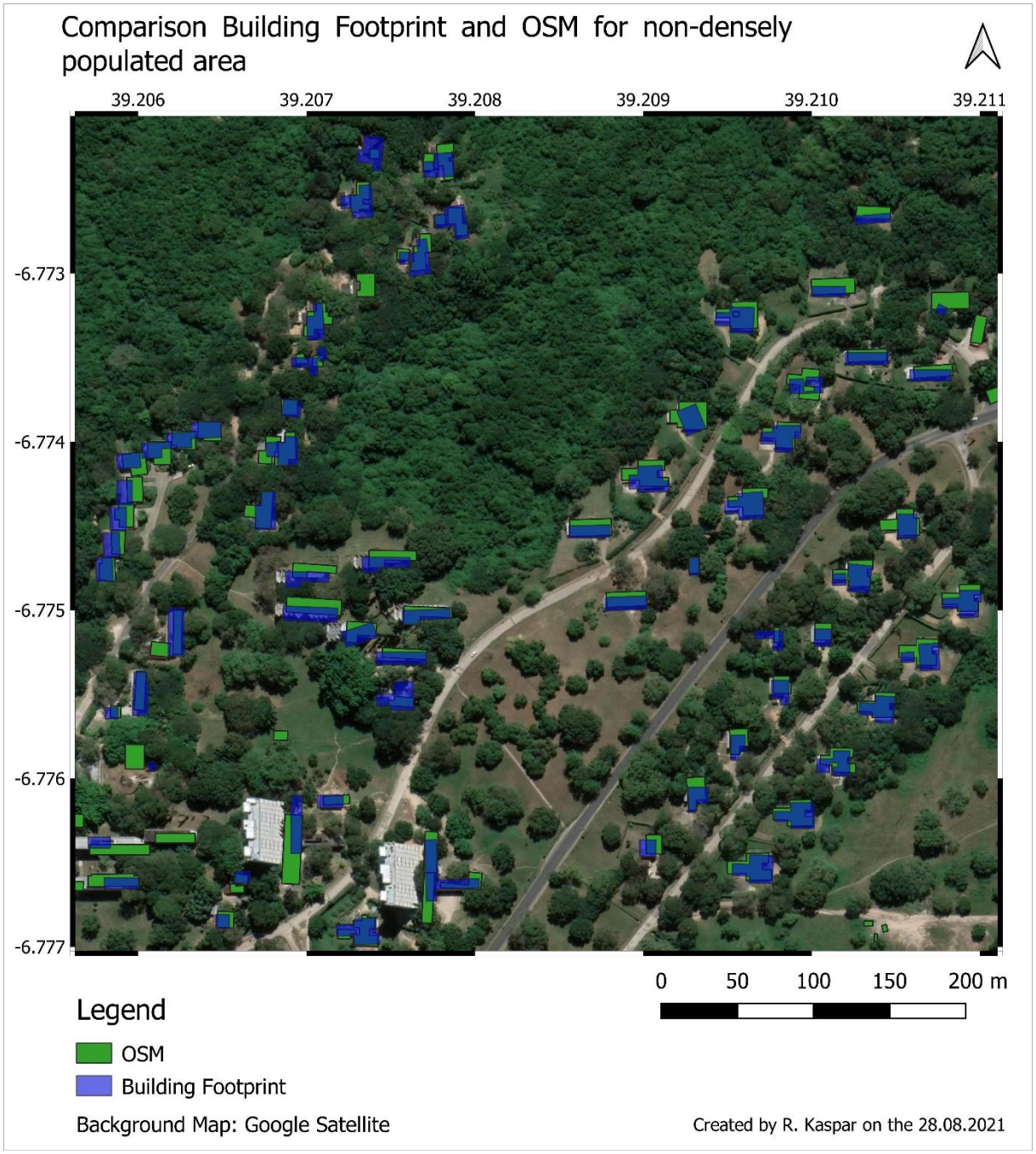

Figure 11: Comparison Building Footprint and OSM for non-densely populated area, Background Map: Google Satellite

The established workflow does not provide a detailed damage assessment, however the calculated numbers and the generated maps could be useful at giving a first overview to decision makers. The maps and indicators can improve monitoring and planning of activities after a flood event. There are several limiting factors in urban areas, however the results point out that the established workflow would work better for rural areas.

5 Conclusion

A workflow to estimate the affected buildings based on the Building Footprint was successfully established. The selection of the flood mapping algorithm, satellite image availability and parameter configuration have a high influence on the result. The flood zone algorithm and parameterisation need to be considered carefully, as this is the major limiting factor in the methodology. Despite the possible issues in the flood detection algorithm, which can arise from a variety of sources, this thesis shows that the Building Footprint can be a useful data source for assessing the level of damage due to a flood event. The results for urban areas are mixed, the number of affected buildings from the Building Footprint are not persuasive, a better indicator is the total area covered by the affected buildings. The Building Footprint results in a slight overestimation of the total area covered by the affected buildings in urban areas. The overestimation is small and can be explained, therefore the indicator can be considered valid for the selected flood event. For the selected study site, the indicator results from OSM data are more suitable, due to higher accuracy of the OSM building perimeters. Presumably for urban areas in general, the OSM data should be considered as an alternative data source. It can be concluded that the biggest potential of the established workflow and the Building Footprint data are within rural areas. This is due to the fact that the flood detection algorithm is less sensitive to errors and the data accuracy of the Building Footprint is much higher compared to urban areas.

6 References

Agnihotri, A. K., Ohri, A., Gaur, S., Shivam, Das, N., & Mishra, S. (2019). Flood inundation mapping and monitoring using SAR data and its impact on Ramganga River in Ganga basin. *Environmental Monitoring and Assessment*, *191*(12), 760. https://doi.org/10.1007/s10661-019-7903-4

Anusha, N., & Bharathi, B. (2019). Flood detection and flood mapping using multi-temporal synthetic aperture radar and optical data. *The Egyptian Journal of Remote Sensing and Space Sciences*, *23*. https://doi.org/10.1016/j.ejrs.2019.01.001

Barnes, C., Fritz, H., & Yoo, J. (2007). Hurricane Disaster Assessments With Image-Driven Data Mining in High-Resolution Satellite Imagery. *IEEE Transactions on Geoscience and Remote Sensing,* 1631–1640. https://doi.org/10.1109/TGRS.2007.890808

Barron, C., Neis, P., & Zipf, A. (2014). A Comprehensive Framework for Intrinsic OpenStreetMap Quality Analysis. *Transactions in GIS*, *18*(6), 877–895. https://doi.org/10.1111/tgis.12073

Braun, B., & Aßheuer, T. (2011). Floods in megacity environments: Vulnerability and coping strategies of slum dwellers in Dhaka/Bangladesh. *Natural Hazards*, *58*(2), 771–787. https://doi.org/10.1007/s11069-011-9752-5

Cao, H., Zhang, H., Wang, C., & Zhang, B. (2019). Operational Flood Detection Using Sentinel-1 SAR Data over Large Areas. *Water*, *11*(4), 786. https://doi.org/10.3390/w11040786

Daily News. (2020). *Heavy rain pounds, paralyses Dar environs.* Retrieved April 10, 2021 from: https://dailynews.co.tz/news/2020-10-135f85f187afafd.aspx

DeVries, B., Huang, C., Armston, J., Huang, W., Jones, J. W., & Lang, M. W. (2020). Rapid and robust monitoring of flood events using Sentinel-1 and Landsat data on the Google Earth Engine. *Remote Sensing of Environment*, *240*, 111664. https://doi.org/10.1016/j.rse.2020.111664

ESA. (2020). *Copernicus Open Access Hub*. Retrieved July 15, 2021 from: https://scihub.copernicus.eu/dhus/#/home

Fernandez-Galarreta, J., Kerle, N., & Gerke, M. (2015). UAV-based urban structural damage assessment using object-based image analysis and semantic reasoning. *Natural Hazards and Earth System Sciences*, *15*, 1087–1101. https://doi.org/10.5194/nhess-15-1087-2015

Ferreira, S., Hamilton, K., & Vincent, J. R. (2011). *Nature, Socioeconomics and Adaptation to Natural Disasters: New Evidence from Floods*. The World Bank. https://doi.org/10.1596/1813-9450-5725

Filipponi, F. (2019). Sentinel-1 GRD Preprocessing Workflow. *Proceedings*, *18*(1), 11. https://doi.org/10.3390/ECRS-3-06201

Geudtner, D., Torres, R., Snoeij, P., Davidson, M., & Rommen, B. (2014). Sentinel-1 System capabilities and applications. *2014 IEEE Geoscience and Remote Sensing Symposium*, 1457–1460. https://doi.org/10.1109/IGARSS.2014.6946711

Giustarini, L., Hostache, R., Matgen, P., Schumann, G. J.-P., Bates, P. D., & Mason, D. C. (2013). A Change Detection Approach to Flood Mapping in Urban Areas Using TerraSAR-X. *IEEE Transactions on Geoscience and Remote Sensing*, *51*(4), 2417–2430. https://doi.org/10.1109/TGRS.2012.2210901

Google Developers. (2008). *WWF HydroSHEDS Void-Filled DEM, 3 Arc-Seconds*. Google Developers. Retrieved July 28, 2021 from: https://developers.google.com/earth-engine/datasets/catalog/WWF_HydroSHEDS_03VFDEM

Google Developers. (2020). *Sentinel-1 SAR GRD: C-band Synthetic Aperture Radar Ground Range Detected, log scaling*. Google Developers. Retrieved July 28, 2021 from: https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S1_GRD

Google Developers. (2021). *JRC Global Surface Water Mapping Layers, v1.3*. Google Developers. Retrieved July 28, 2021 from: https://developers.google.com/earth-engine/datasets/catalog/JRC_GSW1_3_GlobalSurfaceWater

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, *202*, 18–27. https://doi.org/10.1016/j.rse.2017.06.031

Hallegatte, S., Green, C., Nicholls, R. J., & Corfee-Morlot, J. (2013). Future flood losses in major coastal cities. *Nature Climate Change*, *3*(9), 802–806. https://doi.org/10.1038/nclimate1979

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Herfort, B., Lautenbach, S., Porto de Albuquerque, J., Anderson, J., & Zipf, A. (2021). The evolution of humanitarian mapping within the OpenStreetMap community. *Scientific Reports*, *11*(1). https://doi.org/10.1038/s41598-021-82404-z

Heris, M. P., Foks, N. L., Bagstad, K. J., Troy, A., & Ancona, Z. H. (2020). A rasterized building footprint dataset for the United States. *Scientific Data*, *7*(1), 207. https://doi.org/10.1038/s41597-020-0542-3

Huang, M., & Jin, S. (2020). Rapid Flood Mapping and Evaluation with a Supervised Classifier and Change Detection in Shouguang Using Sentinel-1 SAR and Sentinel-2 Optical Data. *Remote Sensing*, *12*(13), 2073. https://doi.org/10.3390/rs12132073

IPCC. (2018). *Global Warming of 1.5°C. An IPCC Special Report on the impacts of global warming of 1.5°C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty [Masson-Delmotte, V., P. Zhai, H.-O. Pörtner, D. Roberts, J. Skea, P.R. Shukla, A. Pirani, W. Moufouma-Okia, C. Péan, R. Pidcock, S. Connors, J.B.R. Matthews, Y. Chen, X. Zhou, M.I. Gomis, E. Lonnoy, T. Maycock, M.Tignor, and T. Waterfield (eds.)]. In Press.*

IPCC. (2021). *Summary for Policymakers. In: Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change [Masson-Delmotte, V., P. Zhai, A. Pirani, S. L. Connors, C. Péan, S. Berger, N. Caud, Y. Chen, L. Goldfarb, M. I. Gomis, M. Huang, K. Leitzell, E. Lonnoy, J.B.R. Matthews, T. K. Maycock, T. Waterfield, O. Yelekçi, R. Yu and B. Zhou (eds.)]. In Press.*

Jiang, B. (2019). Natural Cities Generated from All Building Locations in America. *Data*, *4*(2), 59. https://doi.org/10.3390/data4020059

Jiménez-Jiménez, S. I., Ojeda-Bustamante, W., Ontiveros-Capurata, R. E., & Marcial-Pablo, M. D. J. (2020). Rapid urban flood damage assessment using high resolution remote sensing data and an object-based approach. *Geomatics, Natural Hazards and Risk*, *11*(1), 906–927. https://doi.org/10.1080/19475705.2020.1760360

Jovel, R. J., & Mudahar, M. (2010). *Damage, Loss, and Needs Assessment Guidance Notes: Volume 1. Design and Execution of a Damage, Loss, and Needs Assessment.* Washington, DC: World Bank.

Kabanda, T. (2018). Long-Term Rainfall Trends over the Tanzania Coast. *Atmosphere*, *9*(4), 155. https://doi.org/10.3390/atmos9040155

Kakooei, M., & Baleghi, Y. (2017). Fusion of satellite, aircraft, and UAV data for automatic disaster damage assessment. *International Journal of Remote Sensing*, *38*(8–10), 2511–2534. https://doi.org/10.1080/01431161.2017.1294780

Kordelas, G., Manakos, I., Aragonés, D., Díaz-Delgado, R., & Bustamante, J. (2018). Fast and Automatic Data-Driven Thresholding for Inundation Mapping with Sentinel-2 Data. *Remote Sensing*, *10*(6), 910. https://doi.org/10.3390/rs10060910

Lehner, B., Verdin, K., & Jarvis, A. (2008). New Global Hydrography Derived From Spaceborne Elevation Data. *Eos, Transactions American Geophysical Union*, *89*(10), 93. https://doi.org/10.1029/2008EO100001

Lillesand, T. M., Kiefer, R. W., & Chipman, J. W. (2008). *Remote sensing and image interpretation* (6. ed). Hoboken, NJ: Wiley.

Malinowski, R., Groom, G., Schwanghart, W., & Heckrath, G. (2015). Detection and Delineation of Localized Flooding from WorldView-2 Multispectral Data. *Remote Sensing*, *7*(11), 14853–14875. https://doi.org/10.3390/rs71114853

Manavalan, R., Rao, Y. S., & Krishna Mohan, B. (2017). Comparative flood area analysis of C-band VH, VV, and L-band HH polarizations SAR data. *International Journal of Remote Sensing*, *38*(16), 4645–4654. https://doi.org/10.1080/01431161.2017.1325534

Manjusree, P., Kumar, L. P., Bhatt, C. M., Rao, G. S., & Bhanumurthy, V. (2012). Optimization of threshold ranges for rapid flood inundation mapping by evaluating backscatter profiles of high incidence angle SAR images. *International Journal of Disaster Risk Science*, *3*(2), 113–122. https://doi.org/10.1007/s13753-012-0011-5

Mason, D. C., Dance, S. L., & Cloke, H. L. (2021). Floodwater detection in urban areas using Sentinel-1 and WorldDEM data. *Journal of Applied Remote Sensing*, *15*(3). https://doi.org/10.1117/1.JRS.15.032003

Mason, D. C., Giustarini, L., Garcia-Pintado, J., & Cloke, H. L. (2014). Detection of flooded urban areas in high resolution Synthetic Aperture Radar images using double scattering. *International Journal of Applied Earth Observation and Geoinformation*, *28*, 150–159. https://doi.org/10.1016/j.jag.2013.12.002

Meyer, F. (2019). *Spaceborne Synthetic Aperture Radar: Principles, Data Access, and Basic Processing Techniques. SAR Handbook: Comprehensive Methodologies for Forest Monitoring and Biomass Estimation. Eds. Flores, A., Herndon, K., Thapa, R., Cherrington, E. NASA.* https://doi.org/10.25966/EZ4F-MG98

Microsoft. (2018). United States Building Footprints. *Microsoft Releases 125 Million Building Footprints in the US as Open Data*. Retrieved April 21, 2021, from: https://github.com/Microsoft/USBuildingFootprints

Microsoft. (2019). Uganda Tanzania Building Footprints. *Microsoft Releases 18 Million Building Footprints in Uganda and Tanzania to Enable AI Assisted Mapping*. Retrieved April 21, 2021, from: https://github.com/microsoft/Uganda-Tanzania-Building-Footprints

Moreira, A., Prats-Iraola, P., Younis, M., Krieger, G., Hajnsek, I., & Papathanassiou, K. P. (2013). A tutorial on synthetic aperture radar. *IEEE Geoscience and Remote Sensing Magazine*, *1*(1), 6–43. https://doi.org/10.1109/MGRS.2013.2248301

NBS Tanzania. (2012). *Level three shapefiles of Tanzania: Ward boundaries*. Retrieved July 23, 2021 from: https://www.nbs.go.tz/index.php/en/census-surveys/gis/386-2012-phc-shapefiles-level-three

Notti, D., Giordan, D., Caló, F., Pepe, A., Zucca, F., & Galve, J. (2018). Potential and Limitations of Open Satellite Data for Flood Mapping. *Remote Sensing*, *10*(11), 1673. https://doi.org/10.3390/rs10111673

OCHA. (2016). *2012 Census population by ward level*. Retrieved July 10, 2021 from: https://data.humdata.org/dataset/population-by-ward-adm3

Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, *9*(1), 62–66. https://doi.org/10.1109/TSMC.1979.4310076

Pekel, J.-F., Cottam, A., Gorelick, N., & Belward, A. S. (2016). High-resolution mapping of global surface water and its long-term changes. *Nature*, *540*(7633), 418–422. https://doi.org/10.1038/nature20584

Rahman, M. S., & Di, L. (2017). The state of the art of spaceborne remote sensing in flood management. *Natural Hazards*, *85*(2), 1223–1248. https://doi.org/10.1007/s11069-016-2601-9

Raifer, M. (2021). *Overpass Turbo: A web based data mining tool for OpenStreetMap using the Overpass API*. Retrieved July 4, 2021 from: https://github.com/tyrasd/overpass-turbo

Rana, V. K., & Suryanarayana, T. M. V. (2019). Evaluation of SAR speckle filter technique for inundation mapping. *Remote Sensing Applications: Society and Environment*, *16*, 100271. https://doi.org/10.1016/j.rsase.2019.100271

Scholz, S., Knight, P., Eckle, M., Marx, S., & Zipf, A. (2018). Volunteered Geographic Information for Disaster Risk Reduction—The Missing Maps Approach and Its Potential within the Red Cross and Red Crescent Movement. *Remote Sensing*, *10*(8), 1239. https://doi.org/10.3390/rs10081239

Schuegraf, P., & Bittner, K. (2019). Automatic Building Footprint Extraction from Multi-Resolution Remote Sensing Images Using a Hybrid FCN. *ISPRS International Journal of Geo-Information*, *8*(4), 191. https://doi.org/10.3390/ijgi8040191

Schumann, G. J.-P., & Moller, D. K. (2015). Microwave remote sensing of flood inundation. *Physics and Chemistry of the Earth, Parts A/B/C*, *83–84*, 84–95. https://doi.org/10.1016/j.pce.2015.05.002

Sehra, S., Singh, J., & Rai, H. S. (2017). Using Latent Semantic Analysis to Identify Research Trends in OpenStreetMap. *ISPRS International Journal of Geo-Information*, *6*(7), 195. https://doi.org/10.3390/ijgi6070195

Shen, X., Wang, D., Mao, K., Anagnostou, E., & Hong, Y. (2019). Inundation Extent Mapping by Synthetic Aperture Radar: A Review. *Remote Sensing*, *11*(7), 879. https://doi.org/10.3390/rs11070879

Shu, Z., Hu, X., & Sun, J. (2018). Center-Point-Guided Proposal Generation for Detection of Small and Dense Buildings in Aerial Imagery. *IEEE Geoscience and Remote Sensing Letters*, *15*(7), 1100–1104. https://doi.org/10.1109/LGRS.2018.2822760

Tan, M., & Le, Q. V. (2020). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. https://arxiv.org/abs/1905.11946v5

TheCitizen. (2020). *Heavy rains leave 12 dead in Dar es Salaam.* The Citizen. Retrieved April 10, 2021 from: https://www.thecitizen.co.tz/tanzania/news/heavy-rains-leave-12-dead-in-dar-es-salaam-2718074

Tiwari, V., Kumar, V., Matin, M. A., Thapa, A., Ellenburg, W. L., Gupta, N., & Thapa, S. (2020). Flood inundation mapping- Kerala 2018; Harnessing the power of SAR, automatic threshold detection method and Google Earth Engine. *PLOS ONE*, *15*(8). https://doi.org/10.1371/journal.pone.0237324

TMA. (2020). *Rainfall data, Agrometeorological Database by the Tanzania Meteorological Agency*. Retrieved April 10, 2021 from: http://agromet.meteo.go.tz/rainfall-grids

United Nations. (2019). *World Urbanization Prospects: The 2018 Revision.* New York: United Nations, Department of Economic and Social Affairs, Population Division.

United Nations Human Settlements Programme. (2014). *The state of African cities, 2014: Re-imagining sustainable urban transitions*. Nairobi: UN-Habitat.

UN-SPIDER. (2020). *Recommended Practice: Flood Mapping and Damage Assessment Using Sentinel-1 SAR Data in Google Earth Engine*. Retrieved April 17, 2021 from: https://www.un-spider.org/advisory-support/recommended-practices/recommended-practice-google-earth-engine-flood-mapping

Winsemius, H. C., Jongman, B., Veldkamp, T. I. E., Hallegatte, S., Bangalore, M., & Ward, P. J. (2018). Disaster risk, climate change, and poverty: Assessing the global exposure of poor

people to floods and droughts. *Environment and Development Economics*, *23*(3), 328–348. https://doi.org/10.1017/S1355770X17000444

World Bank. (2016). *Ramani Huria: The atlas of flood resilience in Dar es Salaam. Washington, DC: World Bank*. Retrieved April 17, 2021 from: http://documents1.worldbank.org/curated/en/200421524092301920/pdf/Ramani-Huria-the-atlas-of-flood-resilience-in-Dar-es-Salaam.pdf

Zhang, M., Chen, F., Liang, D., Tian, B., & Yang, A. (2020). Use of Sentinel-1 GRD SAR Images to Delineate Flood Extent in Pakistan. *Sustainability*, *12*(14), 5784. https://doi.org/10.3390/su12145784

Zhao, W., Persello, C., & Stein, A. (2021). Building outline delineation: From aerial images to polygons with an improved end-to-end learning framework. *ISPRS Journal of Photogrammetry and Remote Sensing*, *175*, 119–131. https://doi.org/10.1016/j.isprsjprs.2021.02.014

7 Appendix

7.1 Configuration of Parameters

The following section explains in detail the configuration of the script on the basis of polarization VV and threshold 1.25. It shows the configuration used to receive the results in this thesis. All configurations are conducted in the console script *Flood Map/Indicator*, which manages the overall procedure (see whole script in section 7.2).

Line 9 defines the area of the study site. The study site must be a shapefile saved on a local directory of the computer and thus the file path must be specified. Line 10 sets the path to the Building Footprint. This needs to be a “shapefile copy” of the original GeoJSON file saved locally (see section 2.4). Line 10 determines the output path. All results and intermediate results will be saved in this folder. The script does not create a new folder, hence the folder has to exist already.

```
9.    indirstudy =
'C:\\DokumenteKAR\\privat\\GIS\\UNIGIS\\Masterthesis\\FINALDATA\\StudyArea\\Study_Area_v3.shp'
10.   indirbuildingfootprint =
'C:\\DokumenteKAR\\privat\\GIS/UNIGIS\\Masterthesis\\FINALDATA\\Tanzania_2019-09-
16\\Tanzania_2019-09-16.shp'
11.   outdir =
'C:\\DokumenteKAR\\privat\\GIS\\UNIGIS\\Masterthesis\\FINALDATA\\Export\\20201013VVthreshold125'
```

Line 21 and 22 set the start and end date for the time period for the preflood image or images. It is possible to set a longer time period, as the processing script *Create Flood Map* is using the GEE method *mosaic,* which composites overlapping images with the order the latest on top. Line 21 and 22 define the start and end date for the time period for the after-flood image. The date of the received image should be as close as possible to the flood event. If the script returns several images for the determined flood period the dates should be modified. Figure 12 shows the python console messages after running the console script. The messages include metainformation for the before and after flood images, specifically the number of images and date of the images (see Figure 12 line 6-9). That information simplifies the process of potential time period corrections.

```
21.   'BEFORESTART':'2020-09-01',\
22.   'BEFOREEND':'2020-09-20',\
23.   'AFTERSTART':'2020-10-10',\
24.   'AFTEREND':'2020-10-17',\
```

Line 25 configures the pass direction of the satellite. The parameter can either be set on ASCENDING or DESCENDING. As mentioned in section 2.3 only an ascending pass direction is covering the selected study site completely.

```
25.   'PASSDIRECTION':'ASCENDING',\
```

Line 26 defines the parameter polarization. The methodology allows to select either VV or VH. Line 27 determines the threshold for the change detection approach. Theoretically any floating-point number is possible.

```
26.    'POLARIZATION':'VV',\
27.    'THRESHOLD':1.25,\
```

After running the script, the console messages inform the user about the progress in the procedure. The script is divided in 6 main steps, details in Figure 2 (numbers 2 to 7).

```
Count Images "Before": 2
Dates of Images Before: ['2020-09-07', '2020-09-19']
Count Images "After": 1
Dates of Images After: ['2020-10-13']
Generating URL ...
Downloading data from https://earthengine.googleapis.com/vlalpha/projects/earthengine-legacy/tables/3lb3e680160e19del75ebb2d66325fc7-4bdle3e76lca95a57856a24a5le3ecc7:getFeatures
Please wait ...
Data downloaded to C:\DokumenteKAR\privat\GIS\UNIGIS\Masterthesis\FINALDATA\Export\20201013VVthreshold125\floodedareageemap.shp
Generating URL ...
Downloading data from https://earthengine.googleapis.com/vlalpha/projects/earthengine-legacy/thumbnails/9770ca004f4ae60298287e9d6df85f6l-ldl75cl994d0352be6c88adbaea67fa9:getPixels
Please wait ...
Data downloaded to C:\DokumenteKAR\privat\GIS\UNIGIS\Masterthesis\FINALDATA\Export\20201013VVthreshold125\floodedarea.tif

1/6 steps done. Flood map created. Start fetching OSM data

2/6 steps done. OSM layer saved. Start extracting building footprint.

3/6 steps done. Building footprint for study area extracted

4/6 steps done. Affected buidlings detected

5/6 steps done. Indicators calculated

6/6 steps done. Layers added to map
```

Figure 12: Python console messages after running the console script Flood Map/Indicator

7.2 Python Console Script

```
1.  import processing
2.  import csv
3.
4.  ###--------------------------###
5.  # INIT: Please adapt paths for Study Site, Building Footprint & and output location if nec-
    essary
6.  # Set paths for files (Study area, Building footprint Tanzania, Output location)
7.  ###--------------------------###
8.
9.  indirstudy =
    'C:\\DokumenteKAR\\privat\\GIS\\UNIGIS\\Masterthesis\\FINALDATA\\StudyArea\\Study_Area_v1.sh
    p'
10. indirbuildingfootprint =
    'C:\\DokumenteKAR\\privat\\GIS/UNIGIS\\Masterthesis\\FINALDATA\\Tanzania_2019-09-
    16\\Tanzania_2019-09-16.shp'
11. outdir =
    'C:\\DokumenteKAR\\privat\\GIS\\UNIGIS\\Masterthesis\\FINALDATA\\Export\\20201013VVthreshold
    125'
12.
13. ###--------------------------###
14. # Part 1: Run Processing Script "Create Flood Map" to determine flood area in study area and
    result is saved locally.
15. # Please adapt dates for BEFORE and AFTER flood event. The efective dates of the images are
    printed in the console. One image only for after the flood is preferable.
16. # Depending on the area the passdirection could be set on DESCENDING. Polarization and
    Threshold can be changed, but only for advanced users.
17. ###--------------------------###
18.
19. processing.run("script:createfloodmap",\
20. {'INPUT': indirstudy,\
21. 'BEFORESTART':'2020-09-01',\
22. 'BEFOREEND':'2020-09-20',\
23. 'AFTERSTART':'2020-10-10',\
24. 'AFTEREND':'2020-10-17',\
25. 'PASSDIRECTION':'ASCENDING',\
26. 'POLARIZATION':'VV',\
27. 'THRESHOLD':1.25,\
28. 'OUTPUT': outdir})
29.
30. print("\n1/6 steps done. Flood map created. Start fetching OSM data")
31.
32. ###--------------------------###
33. # Part 2: Run Processing Script "Fetch OSM" to get current OSM data set for the study
    site.ENABLE_USER_SITE
34. # Note: Depending on the size of the study site this step can take a while.
35. ###--------------------------###
36.
37. outdirgpkg_osm = os.path.join(outdir, 'building_layer_osm.gpkg')
38.
39. processing.run("script:FetchOSM",
40. {'study':indirstudy,
41. 'OsmDownload':'TEMPORARY_OUTPUT',
42. 'Package':outdirgpkg_osm,
43. 'VERBOSE_LOG':False})
44.
45. print("\n2/6 steps done. OSM layer saved. Start extracting building footprint.")
46.
47. ###--------------------------###
48. # Part 3: Extract builing footprint for study area with "clip"
49. ###--------------------------###
50.
51. outdirgpkg_footprint = os.path.join(outdir, 'building_layer_footprint.gpkg')
52.
53. processing.run("native:clip",\
54. {'INPUT': indirbuildingfootprint,\
55. 'OVERLAY': indirstudy,\
56. 'OUTPUT':outdirgpkg_footprint})
```

```
57.
58. print("\n3/6 steps done. Building footprint for study area extracted")
59.
60. ###--------------------------###
61. # Part 4: Extract affected builing for building footprint & OSM
62. # Extractbylocation is used to create datasets for OSM buildings within the floodzone and
    for building footprint buildings within the floodzone
63. ###--------------------------###
64.
65. footprint_affected = os.path.join(outdir, 'building_layer_footprint_affected.gpkg')
66. osm_affected = os.path.join(outdir, 'building_layer_osm_affected.gpkg')
67. flooded_area = os.path.join(outdir, 'floodedarea.shp')
68.
69. try:
70.     os.remove(footprint_affected)
71. except OSError as e:
72.     print("Info: footprint_affected is new")
73.
74. processing.run("native:extractbylocation", \
75. {'INPUT':outdirgpkg_footprint,\
76. 'PREDICATE':[0],\
77. 'INTERSECT':flooded_area,\
78. 'OUTPUT': footprint_affected})
79.
80. try:
81.     os.remove(osm_affected)
82. except OSError as e:
83.     print("Info: building_layer_osm_affected is new")
84.
85. processing.run("native:extractbylocation", \
86. {'INPUT':outdirgpkg_osm,\
87. 'PREDICATE':[0],\
88. 'INTERSECT':flooded_area,\
89. 'OUTPUT': osm_affected})
90.
91. print("\n4/6 steps done. Affected buidlings detected")
92.
93. ###--------------------------###
94. # Part 5: Create CSV File with relevant indicator numbers by "Calculate Indicators" script
95. # First a header is created, after that the numbers are calculated for each layer with the
    "Calculate Indicators" script.
96. # The results are saved in the csv. Adapt the filename of the CSV if necessary, the file is
    saved in the output folder
97. ###--------------------------###
98.
99. resultcsv = os.path.join(outdir, 'indicatorvalues.csv')
100.
101.     try:
102.         os.remove(resultcsv)
103.     except OSError as e:
104.         print("Info: indicatorvalues.csv is new")
105.
106.     #Create header
107.     filecsv = open(resultcsv, "a", newline="")
108.     tupel = ("Filename","Number of polygons (buildings)","Total size of polygons in square
    m2")
109.     writer = csv.writer(filecsv)
110.     writer.writerow(tupel)
111.     filecsv.close()
112.
113.     processing.run("script:CalculateIndicators",\
114.     {'INPUT': osm_affected,\
115.     'OUTPUT': resultcsv})
116.
117.     processing.run("script:CalculateIndicators",\
118.     {'INPUT': footprint_affected,\
119.     'OUTPUT': resultcsv})
120.
121.     #Please note that the calculating process for the layer "Building Footprint" needs a
    lot of time and should only be called in case you want to calculate ratios between Total and
    affected buildings, etc.
```

```
122.     #processing.run("script:Calculate Indicators",\
123.     #{'INPUT': outdirgpkg_footprint,\
124.     #'OUTPUT': resultcsv})
125.
126.     #Please note that the calculating process for the layer "Building OSM" needs a lot of
    time and should only be called in case you want to calculate ratios between Total and af-
    fected buildings, etc.
127.     #processing.run("script:Calculate Indicators",\
128.     #{'INPUT': outdirgpkg_osm,\
129.     #'OUTPUT': resultcsv})
130.
131.     processing.run("script:CalculateIndicators",\
132.     {'INPUT': flooded_area,\
133.     'OUTPUT': resultcsv})
134.
135.     processing.run("script:CalculateIndicators",\
136.     {'INPUT': indirstudy,\
137.     'OUTPUT': resultcsv})
138.
139.     print("\n5/6 steps done. Indicators calculated")
140.
141.     ###-------------------------###
142.     # Part 6: Add layers to map
143.     # This last section loads the layers into the map and formats the layers.
144.     # There are two groups "Flood event" and the "Background". The background grouping as-
    sumes that the 3 layers 'Google Maps', 'OSM Standard' and'Google Sat'are already loaded
145.     ###-------------------------###
146.
147.     #Create the layer groups
148.     roottree = QgsProject.instance().layerTreeRoot()
149.     floodgroup = roottree.addGroup("Flood Event")
150.     backgroundgroup = roottree.addGroup("Background")
151.
152.     #Moves layer into group
153.     def movelayer(groupname,name):
154.         nameLayer = QgsProject.instance().mapLayersByName(name)[0]
155.         myLayer = roottree.findLayer(nameLayer.id())
156.         myClone = myLayer.clone()
157.         parent = myLayer.parent()
158.         groupname.insertChildNode(0, myClone)
159.         parent.removeChildNode(myLayer)
160.
161.     #Add study site
162.     dsmstudyarea = iface.addVectorLayer(indirstudy,"","ogr")
163.     dsmstudyarea.setName('Study Site')
164.     dsmstudyarea.setOpacity(0.2)
165.     movelayer(floodgroup,'Study Site')
166.
167.     #Add flood zone
168.     floodzone = iface.addVectorLayer(flooded_area,"","ogr")
169.     floodzone.setName('Flood Zone')
170.     floodzone.setOpacity(0.7)
171.     floodzone.renderer().symbol().setColor(QColor("red"))
172.     floodzone.triggerRepaint()
173.     iface.layerTreeView().refreshLayerSymbology(floodzone.id())
174.     movelayer(floodgroup,'Flood Zone')
175.
176.     #Add affected buildings of Building Footprint
177.     affectedhousesfootprint=iface.addVectorLayer(footprint_affected, "", "ogr")
178.     affectedhousesfootprint.setName('Affected Buildings Building Footprint')
179.     affectedhousesfootprint.setOpacity(0.7)
180.     affectedhousesfootprint.renderer().symbol().setColor(QColor("blue"))
181.     affectedhousesfootprint.triggerRepaint()
182.     iface.layerTreeView().refreshLayerSymbology(affectedhousesfootprint.id())
183.     movelayer(floodgroup,'Affected Buildings Building Footprint')
184.
185.     #Add affected buildings of OSM
186.     affectedhousesosm=iface.addVectorLayer(osm_affected, "", "ogr")
187.     affectedhousesosm.setName('Affected Buildings OSM')
188.     affectedhousesosm.setOpacity(0.7)
189.     affectedhousesosm.renderer().symbol().setColor(QColor("green"))
```

```
190.      affectedhousesosm.triggerRepaint()
191.      iface.layerTreeView().refreshLayerSymbology(affectedhousesosm.id())
192.      movelayer(floodgroup,'Affected Buildings OSM')
193.
194.      #Move Background layers 'Google Maps', 'OSM Standard' and'Google Sat' into group Back-
    ground
195.      if QgsProject.instance().mapLayersByName('Google Maps'):
196.          movelayer(backgroundgroup,'Google Maps')
197.      if QgsProject.instance().mapLayersByName('OSM Standard'):
198.          movelayer(backgroundgroup,'OSM Standard')
199.      if QgsProject.instance().mapLayersByName('Google Sat'):
200.          movelayer(backgroundgroup,'Google Sat')
201.
202.      #Zoom to study site
203.      zoomLayer = QgsProject.instance().mapLayersByName('Study Site')[0]
204.      iface.setActiveLayer(zoomLayer)
205.      iface.zoomToActiveLayer()
206.
207.      print("\n6/6 steps done. Layers added to map")
```

7.3 Processing Script "Create Flood Map"

```
# -*- coding: utf-8 -*-

"""
***************************************************************************
*                                                                         *
*   This program is free software; you can redistribute it and/or modify  *
*   it under the terms of the GNU General Public License as published by  *
*   the Free Software Foundation; either version 2 of the License, or     *
*   (at your option) any later version.                                   *
*                                                                         *
***************************************************************************
"""

from qgis.PyQt.QtCore import QCoreApplication
from qgis.core import (QgsProcessing,
                       QgsFeatureSink,
                       QgsProcessingException,
                       QgsProcessingAlgorithm,
                       QgsProcessingParameterFeatureSource,
                       QgsProcessingParameterFeatureSink,
                       QgsProcessingParameterFolderDestination,
                       QgsProcessingParameterNumber,
                       QgsProcessingParameterVectorLayer,
                       QgsProcessingParameterString,
                       QgsProcessingParameterFile)
from qgis import processing
import ee
import geemap
import os
from ee_plugin import Map
#Determine Flood Zone from SAR1 Data through GGE
class CreateFloodMap(QgsProcessingAlgorithm):

    INPUT = 'INPUT'
    OUTPUT = 'OUTPUT'

    def tr(self, string):
        return QCoreApplication.translate('Processing', string)

    def createInstance(self):
        return CreateFloodMap()

    def name(self):
        return 'createfloodmap'

    def displayName(self):
        return self.tr('Create Flood Map')

    def group(self):
        return self.tr('MSC')

    def groupId(self):
        return 'MSC'

    def shortHelpString(self):
        return self.tr("Creates Polgyon with flooded area for a specific flood event (SAR 1
    data via GEE Catalog)")

    #Input Parameter definition
    def initAlgorithm(self, config=None):

        self.addParameter(
            QgsProcessingParameterFile(
                self.INPUT,
                self.tr('Study Area (.shp)')
            )
        )
```

```
        self.addParameter(
            QgsProcessingParameterString(
                'BEFORESTART',
                self.tr('Start Date before'),
                defaultValue = 'yyyy-mm-dd'
            )
        )
        self.addParameter(
            QgsProcessingParameterString(
                'BEFOREEND',
                self.tr('End Date before')
            )
        )
        self.addParameter(
            QgsProcessingParameterString(
                'AFTERSTART',
                self.tr('Start Date after')
            )
        )
        self.addParameter(
            QgsProcessingParameterString(
                'AFTEREND',
                self.tr('End Date after')
            )
        )
        self.addParameter(
            QgsProcessingParameterString(
                'PASSDIRECTION',
                self.tr('Pass Direction'),
                defaultValue="ASCENDING"
            )
        )
        self.addParameter(
                 QgsProcessingParameterString(
                     'POLARIZATION',
                     self.tr('Polarization'),
                     defaultValue="VH"
                 )
             )
             self.addParameter(
                 QgsProcessingParameterNumber(
                     'THRESHOLD',
                     self.tr('Threshold'),
                     type=QgsProcessingParameterNumber.Double,
                     defaultValue=1.2
                 )
             )
             self.addParameter(
                 QgsProcessingParameterFolderDestination(
                     self.OUTPUT,
                     self.tr('Output folder')
                 )
             )

         #Processing
         def processAlgorithm(self, parameters, context, feedback):

             #Initialize library.
             ee.Initialize()

             #-------------------------------------------
             # KEY Variables for the flood map
             #-------------------------------------------
             #Timeframe BEFORE the flood.
             before_start= self.parameterAsString(parameters, 'BEFORESTART', context)
             before_end= self.parameterAsString(parameters, 'BEFOREEND', context)

             #Timeframe AFTER the flood.
             after_start= self.parameterAsString(parameters, 'AFTERSTART', context)
             after_end= self.parameterAsString(parameters, 'AFTEREND', context)

```

```
138.             #Set polarization
139.             polarization_primary = self.parameterAsString(parameters, 'POLARIZATION', con-
    text)
140.             #polarization_secondary = "VV"
141.
142.             # Either Ascending or Descending
143.             pass_direction = self.parameterAsString(parameters, 'PASSDIRECTION', context)
144.
145.             #Threshold for pixel classificaton into flood/Non flood
146.             difference_threshold = self.parameterAsDouble(parameters, 'THRESHOLD', context)
147.
148.             #Path for study area
149.             study_shp = parameters['INPUT']
150.             out_directory = str(parameters['OUTPUT'])
151.
152.             #Minimaly considered area of flood in pixels, only leave areas greater than
    minimal size area
153.             #1 pixel = 10x10 meters
154.             minimalsizearea=10
155.             #Keep only pixels with less than defined value for degrees
156.             degree = 5
157.             #-----------------------------------------------
158.             # End of KEY Variables for the flood Map
159.             #-----------------------------------------------
160.
161.             #Convert study (shp) to Earth Engine object & create FeatureCollection
162.             studyarea = geemap.shp_to_ee(study_shp)
163.             aoi = ee.FeatureCollection(studyarea)
164.
165.             #Function checks number of images in collection
166.             def countimages(imageCollection):
167.                 return imageCollection.size()
168.
169.             #Function evalutes dates of images in collection
170.             def datesimages(imageCollection):
171.                 return imageCollection.map( lambda image : ee.Feature(None, {'date':
    image.date().format('YYYY-MM-dd')}))\
172.                     .distinct('date')\
173.                     .aggregate_array('date')
174.
175.             #For calculating flood area only one polarization is used --> best polarization
    for urban areas according to lterature is VH
176.             collection= ee.ImageCollection('COPERNICUS/S1_GRD')\
177.                 .filter(ee.Filter.eq('instrumentMode','IW'))\
178.                 .filter(ee.Filter.listContains('transmitterReceiverPolarisation', polariza-
    tion_primary))\
179.                 .filter(ee.Filter.eq('orbitProperties_pass',pass_direction)) \
180.                 .filter(ee.Filter.eq('resolution_meters',10))\
181.                 .filterBounds(aoi)\
182.                 .select(polarization_primary)
183.
184.             #Get imagecollection before flood
185.             before_collection = collection.filterDate(before_start, before_end)
186.             #Get imagecollection after flood
187.             after_collection = collection.filterDate(after_start,after_end)
188.
189.             #Mosaic() method composites overlapping images according to their order in the
    collection (last on top)
190.             #clip to aoi --> only works on images --> needs mosaic wouldnt work on im-
    agecollection
191.             before = before_collection.mosaic().clip(aoi)
192.             after = after_collection.mosaic().clip(aoi)
193.
194.             #Apply smoothing to reduce speckle
195.             smoothing_radius = 50
196.             before_filtered_smoothing = before.focal_mean(smoothing_radius, 'circle', 'me-
    ters')
197.             after_filtered_smoothing = after.focal_mean(smoothing_radius, 'circle', 'me-
    ters')
198.
199.             #Images are in Db therefore divide is more accurate
```

```
200.             difference_smoothing = af-
    ter_filtered_smoothing.divide(before_filtered_smoothing)
201.             #Set threshold
202.             threshold = ee.Number(difference_threshold)
203.
204.             #Print number of images and dates of images before flood
205.             countimagesbef = countimages(before_collection).getInfo()
206.             print('Count Images "Before":', countimagesbef)
207.             datesimagesbef = datesimages(before_collection).getInfo()
208.             print('Dates of Images Before:', datesimagesbef)
209.
210.             #Print number of images and dates of images after flood
211.             countimagesaf = countimages(after_collection).getInfo()
212.             print('Count Images "After":', countimagesaf)
213.             datesimagesaf = datesimages(after_collection).getInfo()
214.             print('Dates of Images After:', datesimagesaf)
215.
216.             #Classifies pixel into 0/1 according to threshold (gt=greater than)and selfmask
    removes all 0 values
217.             flooded_difference_binary_smoothing = differ-
    ence_smoothing.gt(threshold).selfMask()
218.
219.             #Define permanent water area in aoi min. 10 months water --> sesonality parame-
    ter
220.             swater = ee.Image('JRC/GSW1_0/GlobalSurfaceWater').select('seasonality')
221.             swater_mask = swater.gte(10).updateMask(swater.gte(10))
222.
223.             #Masking pixels: Pixels are set on transparent and excludes them for further
    process
224.             flooded_mask = flooded_difference_binary_smoothing.where(swater_mask,0)
225.             flooded = flooded_mask.updateMask(flooded_mask)
226.
227.             #Remove isolated pixels
228.             connections = flooded.connectedPixelCount()
229.             #Only leave areas greater than minimalsizearea ---> 1 pixel = 10x10 meters
230.             flooded = flooded.updateMask(connections.gte(minimalsizearea))
231.
232.             #Choose a DEM from GEE Catalog --> Hydrography DEM
233.             DEM = ee.Image('WWF/HydroSHEDS/03VFDEM')
234.             #Calculates slope, hillshade, etc from a DEM
235.             terrain = ee.Algorithms.Terrain(DEM)
236.             #Select slope in degrees from terrain DEM.
237.             slope = terrain.select('slope')
238.             #Update flooded variable and only keep areas with less than 5 degrees
239.             flooded = flooded.updateMask(slope.lt(degree))
240.
241.             #"Backup" of Vector File creation using geemap
242.             #Attention parameter "max Pixels" depend on study area in case of Error it
    needs to be changed
243.             flooded_vector = flooded.reduceToVectors(geometry= aoi.geometry(), scale=10,
    maxPixels = 300000000)
244.             filename_out_geemap = os.path.join(out_directory, 'floodedareageemap.shp')
245.             geemap.ee_export_vector(flooded_vector, filename=filename_out_geemap)
246.
247.             #Save Layer to output folder as TIF
248.             filename_outtif = os.path.join(out_directory, 'floodedarea.tif')
249.             geemap.ee_export_image(flooded, filename=filename_outtif, scale=10, re-
    gion=aoi.geometry(), file_per_band=False)
250.
251.             #Raster to Vector by algorithm polygonize
252.             filename_out_temp = os.path.join(out_directory, 'floodedarea_temp.shp')
253.             try:
254.                 os.remove(filename_out_temp)
255.             except OSError as e:
256.                 print("Info: floodedarea_temp.shp is new")
257.
258.             processing.run("gdal:polygonize",\
259.             {'INPUT':filename_outtif,\
260.             'BAND':1,'FIELD':'DN',\
261.             'EIGHT_CONNECTEDNESS':False,\
262.             'EXTRA':'','OUTPUT': filename_out_temp})
```

```
263.
264.            #To avoid errors due to invalid geometries, use algorithm fixgeometries
265.            filename_out_temp1 = os.path.join(out_directory, 'floodedarea_temp1.shp')
266.            try:
267.                os.remove(filename_out_temp1)
268.            except OSError as e:
269.                print("Info: floodedarea_temp1.shp is new")
270.
271.            processing.run("native:fixgeometries",
272.            {'INPUT': filename_out_temp,
273.            'OUTPUT': filename_out_temp1})
274.
275.            #Clip flood zones to study site by algorithm polygonize
276.            filename_final = os.path.join(out_directory, 'floodedarea.shp')
277.            try:
278.                os.remove(filename_final)
279.
280.            except OSError as e:
281.                print("Info: floodedarea.shp is new")
282.
283.            processing.run("native:clip",\
284.            {'INPUT':filename_out_temp1,\
285.            'OVERLAY':study_shp,\
286.            'OUTPUT': filename_final})
287.
288.            return {}
```

7.4 Model & Processing Script "Fetch OSM"

Figure 13 in section 7.4.1 summarizes the established Model to fetch OSM data. The model was saved as a script and is shown in section 7.4.2 to allow replicability.

7.4.1 Model

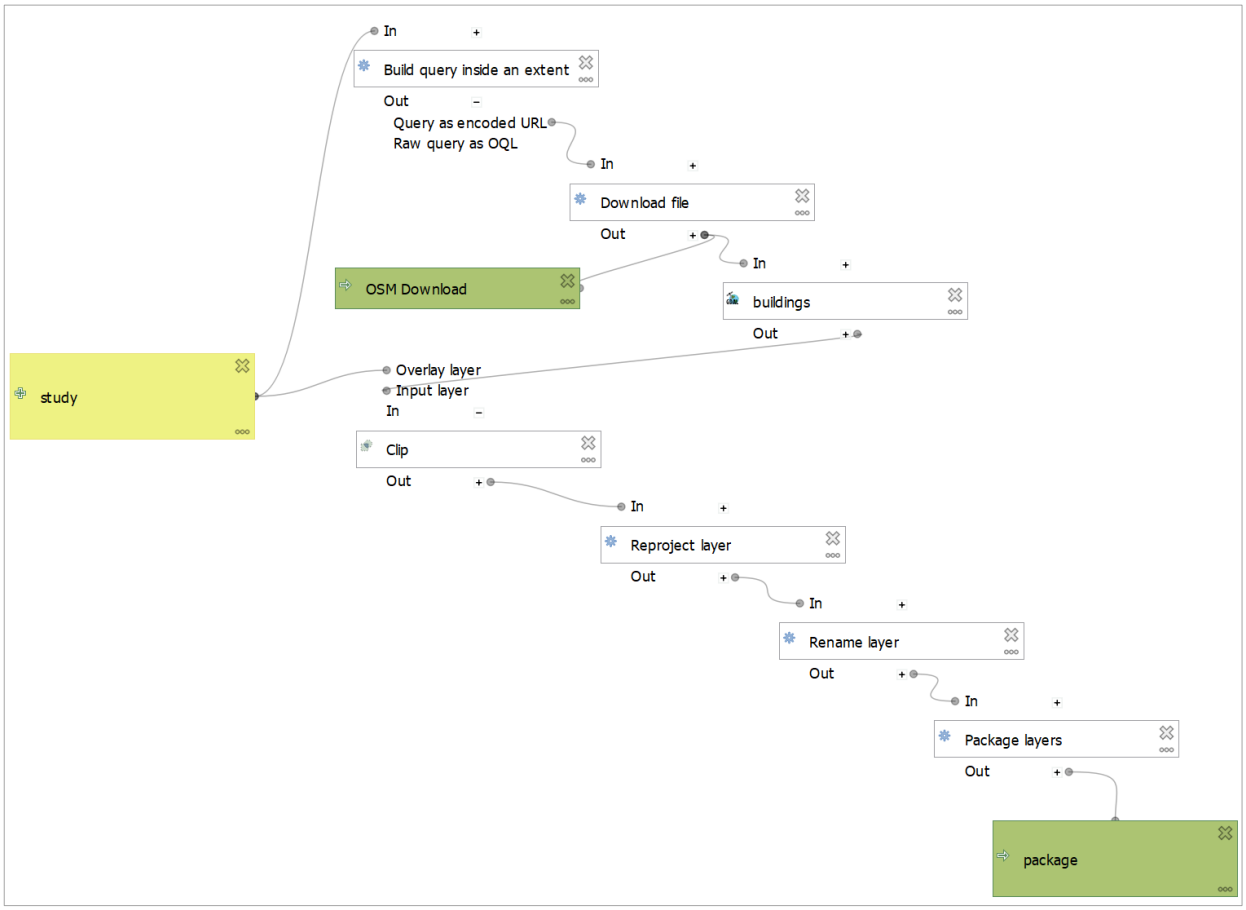

Figure 13: Model Fetch OSM through Overpass Turbo API

7.4.2 Script

```
"""
Model exported as python.
Name : FetchOSM
Group : MSC
With QGIS : 31803
"""

from qgis.core import QgsProcessing
from qgis.core import QgsProcessingAlgorithm
from qgis.core import QgsProcessingMultiStepFeedback
from qgis.core import QgsProcessingParameterVectorLayer
from qgis.core import QgsProcessingParameterFileDestination
from qgis.core import QgsProcessingParameterBoolean
from qgis.core import QgsCoordinateReferenceSystem
import processing


class FetchOSM(QgsProcessingAlgorithm):

```

```
20.     def initAlgorithm(self, config=None):
21.         self.addParameter(QgsProcessingParameterVectorLayer('study', 'study',
    types=[QgsProcessing.TypeVectorPolygon], defaultValue=None))
22.         self.addParameter(QgsProcessingParameterFileDestination('OsmDownload', 'OSM Down-
    load', optional=True, fileFilter='All files (*.*)', createByDefault=True, default-
    Value=None))
23.         self.addParameter(QgsProcessingParameterFileDestination('Package', 'package', file-
    Filter='GeoPackage files (*.gpkg)', createByDefault=True, defaultValue=None))
24.         self.addParameter(QgsProcessingParameterBoolean('VERBOSE_LOG', 'Verbose logging',
    optional=True, defaultValue=False))
25.
26.     def processAlgorithm(self, parameters, context, model_feedback):
27.         # Use a multi-step feedback, so that individual child algorithm progress reports are
    adjusted for the
28.         # overall progress through the model
29.         feedback = QgsProcessingMultiStepFeedback(7, model_feedback)
30.         results = {}
31.         outputs = {}
32.
33.         # Build query inside an extent
34.         alg_params = {
35.             'EXTENT': parameters['study'],
36.             'KEY': 'building',
37.             'SERVER': 'https://lz4.overpass-api.de/api/interpreter',
38.             'TIMEOUT': 250,
39.             'VALUE': ''
40.         }
41.         outputs['BuildQueryInsideAnExtent'] = processing.run('quickosm:buildqueryextent',
    alg_params, context=context, feedback=feedback, is_child_algorithm=True)
42.
43.         feedback.setCurrentStep(1)
44.         if feedback.isCanceled():
45.             return {}
46.
47.         # Download file
48.         alg_params = {
49.             'URL': outputs['BuildQueryInsideAnExtent']['OUTPUT_URL'],
50.             'OUTPUT': parameters['OsmDownload']
51.         }
52.         outputs['DownloadFile'] = processing.run('native:filedownloader', alg_params, con-
    text=context, feedback=feedback, is_child_algorithm=True)
53.         results['OsmDownload'] = outputs['DownloadFile']['OUTPUT']
54.
55.         feedback.setCurrentStep(2)
56.         if feedback.isCanceled():
57.             return {}
58.
59.         # buildings
60.         alg_params = {
61.             'INPUT': outputs['DownloadFile']['OUTPUT'],
62.             'OPTIONS': '-sql \"select * from multipolygons\" -t_srs EPSG:3857',
63.             'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
64.         }
65.         outputs['Buildings'] = processing.run('gdal:convertformat', alg_params, con-
    text=context, feedback=feedback, is_child_algorithm=True)
66.
67.         feedback.setCurrentStep(3)
68.         if feedback.isCanceled():
69.             return {}
70.
71.         # Clip
72.         alg_params = {
73.             'INPUT': outputs['Buildings']['OUTPUT'],
74.             'OVERLAY': parameters['study'],
75.             'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
76.         }
77.         outputs['Clip'] = processing.run('native:clip', alg_params, context=context, feed-
    back=feedback, is_child_algorithm=True)
78.
79.         feedback.setCurrentStep(4)
80.         if feedback.isCanceled():
```

```
            return {}

        # Reproject layer
        alg_params = {
            'INPUT': outputs['Clip']['OUTPUT'],
            'OPERATION': '',
            'TARGET_CRS': QgsCoordinateReferenceSystem('EPSG:4326'),
            'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
        }
        outputs['ReprojectLayer'] = processing.run('native:reprojectlayer', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

        feedback.setCurrentStep(5)
        if feedback.isCanceled():
            return {}

        # Rename layer
        alg_params = {
            'INPUT': outputs['ReprojectLayer']['OUTPUT'],
            'NAME': 'osm_buildings'
             }
             outputs['RenameLayer'] = processing.run('native:renamelayer', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

             feedback.setCurrentStep(6)
             if feedback.isCanceled():
                 return {}

             # Package layers
             alg_params = {
                 'LAYERS': outputs['RenameLayer']['OUTPUT'],
                 'OVERWRITE': True,
                 'SAVE_STYLES': True,
                 'OUTPUT': parameters['Package']
             }
             outputs['PackageLayers'] = processing.run('native:package', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
             results['Package'] = outputs['PackageLayers']['OUTPUT']
             return results

         def name(self):
             return 'FetchOSM'

         def displayName(self):
             return 'Fetch OSM'

         def group(self):
             return 'MSC'

         def groupId(self):
             return 'MSC'

         def createInstance(self):
             return FetchOSM()
```

7.5 Processing Script "Calculate Indicators"

```
# -*- coding: utf-8 -*-

"""
***************************************************************************
*                                                                         *
*   This program is free software; you can redistribute it and/or modify  *
*   it under the terms of the GNU General Public License as published by  *
*   the Free Software Foundation; either version 2 of the License, or     *
*   (at your option) any later version.                                   *
*                                                                         *
***************************************************************************
"""
from qgis.PyQt.QtCore import QVariant
from qgis.core.additions.edit import edit
from qgis.PyQt.QtCore import QCoreApplication
from qgis.core import (QgsProcessing,
                       QgsFeatureSink,
                       QgsProcessingException,
                       QgsProcessingAlgorithm,
                       QgsProcessingParameterFeatureSource,
                       QgsProcessingParameterFeatureSink,
                       QgsProcessingParameterFileDestination,
                       QgsVectorLayer,
                       QgsProcessingFeatureSource,
                       QgsField,
                       QgsExpression,
                       QgsExpressionContext,
                       QgsExpressionContextUtils)
from qgis import processing
import csv

# Calculate indicator values and save into the csv file
class CalculateIndicators(QgsProcessingAlgorithm):

    INPUT = 'INPUT'
    OUTPUT = 'OUTPUT'

    def tr(self, string):
        return QCoreApplication.translate('Processing', string)

    def createInstance(self):
        return CalculateIndicators()

    def name(self):
        return 'CalculateIndicators'

    def displayName(self):
        return self.tr('Calculate Indicators')

    def group(self):
        return self.tr('MSC')

    def groupId(self):
        return 'MSC'

    def shortHelpString(self):
        return self.tr("Calculates the Sum of the Features within the layer")

    #Input Parameter definition
    def initAlgorithm(self, config=None):

        self.addParameter(
            QgsProcessingParameterFeatureSource(
                self.INPUT,
                self.tr('Input layer'),
                [QgsProcessing.TypeVectorAnyGeometry]
            )
```

```
        )

        self.addParameter(
            QgsProcessingParameterFileDestination(
                self.OUTPUT,
                self.tr('Output file'),
                'CSV files(*.csv)',
            )
        )

    #Processing
    def processAlgorithm(self, parameters, context, feedback):

        #Set output directory
        resultcsv = parameters['OUTPUT']

        #Input dataset for the indicator calculation
        source = parameters['INPUT']

        #Define input source dataset as vector layer
        layer = QgsVectorLayer(source, '', 'ogr')

        #Counts number of poygons in layer
        featscount = layer.featureCount()
        feedback.pushInfo(str(featscount))

        #Set connection to layer and add Attribute "Calc1" with type double
        prov = layer.dataProvider()
        prov.addAttributes([QgsField('calc1', QVariant.Double)])
        layer.updateFields()

        #Define expressions for calculation in each field
             expressionarea = QgsExpression('$area')

             #Define relevant layer to perform calculation
             context = QgsExpressionContext()
             context.appendScopes(QgsExpressionContextUtils.globalProjectLayerScopes(layer))

             #Edit layer and calculate for each feature in attribute calc1 the area
             with edit(layer):
                 for f in layer.getFeatures():
                     context.setFeature(f)
                     f['calc1'] = expressionarea.evaluate(context)
                     layer.updateFeature(f)

             #Calculate sum of field calc1 from all features
             sumofarea= sum(filter(None,[f['calc1'] for f in layer.getFeatures()]))
             feedback.pushInfo(str(sumofarea))

             #Export layername, number of feature and sum of the area of all feature into
    csv
             filecsv = open(resultcsv, "a", newline="")
             tupel = (source, featscount, sumofarea)
             writer = csv.writer(filecsv)
             writer.writerow(tupel)
             filecsv.close()

             return {}
```

7.6 System Setup

The scripts and processes in the methodology were created with QGIS Version 3.20.1-Odense and Python Version 3.9.5. The script needs the package GDAL, hence it needs to be installed during the QGIS installation process.

Integrating Google Earth Engine into QGIS relies on the plugin *Google Earth Engine*. The plugin can be installed via *Manage and Install Plugins*. The first time the plugin is started, it requires an authorization token, which can be received from a personal google account.

The processing scripts *Create Flood Map* is using geemap, a python package for interactive mapping with Google Earth Engine and helps to simplify requests for the Earth Engine servers. Geemap needs to be installed on the computer, hence the following command need to be run as administrator in the OSGeo4W Shell.

```
Pip install geemap
```

Low threshold (for example 1.10) can sporadically lead to invalid geometry polygons for the flood zones. To avoid errors during the *extract by location* algorithm, the settings for *Invalid features* filtering is set on *Skip features with invalid geometries* under Options in the processing toolbox.

The processing scripts *Fetch OSM* depends on the plugin *QuickOSM.* The plugin can be installed via *Manage and Install Plugins*. The plugin allows to download OSM data through the Overpass API.

7.6.1 Background Map

```
#Load Basemaps as background

#Create group Background
roottree = QgsProject.instance().layerTreeRoot()

#Load BackgroundMap into layers
def LoadBackgroundMap(url, name):
    rasterLyr = QgsRasterLayer("type=xyz&url=" + url, name, "wms")
    QgsProject.instance().addMapLayer(rasterLyr, False)
    roottree.addLayer(rasterLyr)
    roottree.findLayer(rasterLyr.id()).setItemVisibilityChecked(False)

#Add Basemaps
linkbasemap = 'type=xyz&url=https://mt1.google.com/vt/lyrs%3Dm%26x%3D%7Bx%7D%26y%3D%7By%7D%26z%3D%7Bz%7D&zmax=19&zmin=0'
LoadBackgroundMap(linkbasemap, 'Google Maps')
linkbasemap = 'type=xyz&url=https://mt1.google.com/vt/lyrs%3Ds%26x%3D%7Bx%7D%26y%3D%7By%7D%26z%3D%7Bz%7D&zmax=19&zmin=0'
LoadBackgroundMap(linkbasemap, 'Google Sat')
linkbasemap = 'referer=OpenStreetMap%20contributors,%20under%20ODbL&type=xyz&url=http://tile.openstreetmap.org/%7Bz%7D/%7Bx%7D/%7By%7D.png&zmax=19&zmin=0'
LoadBackgroundMap(linkbasemap, 'OSM Standard')

#Set one Background Map "Google Maps" on visible
enableLayer = QgsProject.instance().mapLayersByName('Google Maps')[0]
roottree.findLayer(enableLayer.id()).setItemVisibilityChecked(True)
```