Bachelor Thesis

Ein offenes Werkzeug zur Messung der räumlichen Zersiedelung

Ein offenes Werkzeug für das Berechnen der räumlichen Zersiedlung

Joël Schwab, Ryan Horiguchi

10. Juni 2020

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

This document was typeset using LaTeX and KOMA-Script and the HSRReport style by H.Badertscher.
Rapperswil, 10. Juni 2020

Aufgabenstellung

Bachelorarbeit im Frühlingssemester 2020, Studiengang Informatik

Hintergrund

Die Zersiedelung ist das Abbild des ungestümen Siedlungswachstums. Es sind vor allem die locker bebauten und dünn besiedelten Gebiete – meist fern des Ortskerns, die das Phänomen sichtbar machen. Die Zersiedelung ist mit der Messgrösse Z nach Schwick und Jäger messbar. Die Sektion Grundlagen des Bundesamtes für Raumentwicklung (ARE) will prüfen, wie diese Messgrösse (Zahl, Indikator) in der Raumplanung eingesetzt werden kann.

Aktueller Stand

Mit der Publikation «Zersiedelung messen und begrenzen» hat die Eidgenössische Forschungsanstalt für Wald, Schnee und Landschaft (WSL) schweizweite Berechnungen durchgeführt und ein Werkzeug online gestellt, das die Berechnungen selber durchführen lässt. Die erforderliche Konfiguration am Desktop GIS ArcGIS, um das «USM -Toolset» laufen zu lassen, ist für den Alltagsgebrauch wenig geeignet. Dies vor allem auch, weil teure Erweiterungen von ArcGIS erforderlich sind.

Aufgabe

Die Zersiedelung kann durch Planende gemessen werden. Als Kern dient hierfür das Open-Source Desktop-GIS QGIS ggf. mit einem Webservice. Das Instrumentarium umfasst ein Werkzeug, das den Planenden ermöglicht, einfach, effizient und zeitnah den Effekt ihrer planerischen Intervention auf die Zersiedelung zu messen. Im Wesentlichen wird das Siedlungsgebiet als Fläche «aufgezeichnet», mit der Anzahl Erwerbstätigen überlagert und die Berechnung ausgelöst. Die Resultate zeigen den Effekt auf, wie die Zersiedelung sich entwickelt.

Umsetzung des Projekts

Die Studierenden müssen selber einschätzen, welche Etappen im Rahmen der Arbeit in welcher Tiefe bearbeitet werden können. Die Arbeit kann wie folgt eingeteilt werden:

- Ausarbeiten der Architektur des Werkzeuges.
- Evaluieren von «Processing as a Service» vs. «Berechnung via QGIS Processing Algorithmus».

- Proof-of-Concept erstellen zum Zersiedelungs-Indikator nach Schwick und Jäger.
- Realisierung des Werkzeuges für die Berechnung der Messgrösse.

Vorgaben – Technologien – Rahmenbedingungen

- Desktop QGIS 3
- Computersprache: Python 3
- Persistenz: GeoPackage, ggf. PostgreSQL

Termine und Bewertungsschema

Es gelten die üblichen Termine und Regelungen zum Ablauf und zur Bewertung der Arbeit (sechs Aspekte) des Studiengangs Informatik der HSR. Zusätzlich wird besonderes Gewicht gelegt auf moderne Softwareentwicklung (Versioning, Continuous Testing & Integration, etc.).

Beteiligte

Diplomanden: Ryan Horiguchi und Joël Schwab

Betreuer: Prof. Stefan Keller, Institut für Software HSR

Weitere Beteiligte: Yves Maure, Bundesamt für Raumentwicklung (ARE)

Abstract

Im Bereich der Geografie ist das Thema der Zersiedelung immer wichtiger. Unter der Zersiedelung versteht man das ungestüme Siedlungswachstum. Dieses Abbild ist meistens ersichtlich, wenn man sich vom Ortskern entfernt. Die Sektion Grundlagen des Bundesamtes für Raumentwicklung (ARE) [1] will überprüfen, wie man den Zersiedelungswert in der Raumplanung einsetzen kann.

Die Berechnung der Zersiedelung wurde bereits von Schwick und Jäger definiert und im Buch 'Zersiedelung messen und begrenzen' [2] beschrieben. Nebst der Literatur gab es ein closed-source Skript, das in C++ geschrieben wurde und die Berechnung der Dispersion zeigte, als Vorlage.

Bisher gab es nur wenige Werkzeuge, mit denen man die Zersiedelung berechnen konnte. Ein solches Tool wurde bereits von der Eidgenössischen Forschungsanstalt für Wald, Schnee und Landschaft (WSL) [3] entwickelt. Für dieses Tool war jedoch ArcGIS [4] erforderlich, welches kommerziell ist. Die Nutzung des Tools erforderte zusätzlich eine spezielle Konfiguration von ArcGIS, die unteranderem teuer war. Dies führte dazu, dass es nur begrenzt nutzbar ist.

Das Ziel ist nun, für das Open Source Tool QGIS [5] eine Erweiterung zu schreiben, die es ermöglicht die Zersiedelung einfach und zeitnahe zu berechnen. Dies erlaubt es dem Benutzer bei Interventionen in ein Siedlungsgebiet die Änderungen festzustellen.

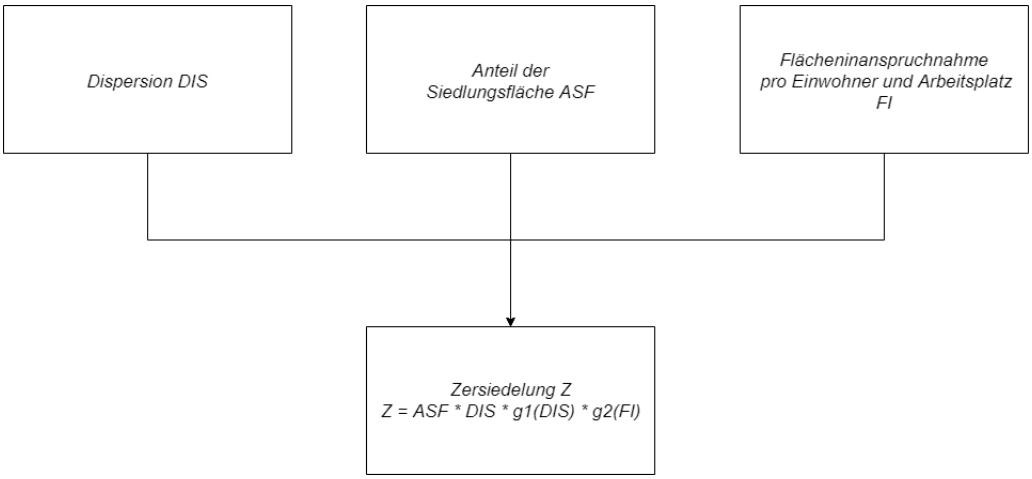
Die Urban Sprawl QGIS Erweiterung ist ein Set von Processing Algorithmen, die es den Benutzern erlauben die Zersiedelung mit einfachen Input Daten durchzuführen. Dadurch, dass es sich um Processing Algorithmen handelt, ist die Berechnung sehr flexibel. Es erlaubt dem Benutzer Schritte einzeln zu berechnen oder alles an einem Stück durchzuführen. Da keine Abhängigkeiten bestehen zwischen den einzelnen Algorithmen, können diese im Falle, dass sich die Berechnung ändert, einfach ausgetauscht werden. Das Tool benutzt keine Parallelisierung in der momentanen Version. Die Berechnung der Zersiedelung einer mittelgrossen Gemeinde in der Schweiz ist trotzdem in akzeptabler Zeit durchführbar.

[1] https://www.are.admin.ch/are/de/home.html
[2] https://www.haupt.ch/Verlag/Buecher/Natur/Umwelt-Oekologie/Zersiedelung-messen-und-begrenzen.html
[3] wsl.ch
[4] https://www.arcgis.com/index.html
[5] https://www.qgis.org/en/site/

Management Summary

Ausgangslage

Die Zersiedelung der Landschaft (englisch 'Urban Sprawl') ist ein Abbild des ungestümen Siedlungswachstums. Es sind dies vor allem die locker bebauten und dünn besiedelten Gebiete – meist fern der Ortskerne, die das Phänomen sichtbar machen. Die Zersiedelung ist mit der Messgrösse Z nach Schwick und Jäger messbar. Als Input dient dabei einerseits die Lage der Gebäude und die Grenzen des Gebiets, andererseits die Anzahl der Erwerbstätigen und die Anzahl der Einwohner eines Untersuchungsgebiets. Der letzte Input ist der Anteil der Siedlungsfläche. Die Sektion Grundlagen des Bundesamtes für Raumentwicklung (ARE)[6] will mit dieser Arbeit prüfen, ob und wie diese Messgrösse Z als Indikator in der kommunalen Raumplanung eingesetzt werden kann. Als Vorlage dient - nebst der theoretischen Literatur - eine proprietäre Closed-Source-Lösung sowie ein in C++ geschriebenes Skript, welches die Dispersion berechnet. Nebst der Herausforderung der Implementation des vorgegebenen mathematischen Modells steht auch die Frage im Raum, ob das Zeitverhalten der neuen Lösung akzeptabel ist, denn bisher dauerten die Berechnungen teilweise Stunden. Im Vordergrund der Zielgruppen stehen einfache Raumplanerbüros. Bei Interventionen in ein Siedlungsgebiet sollen Änderungen einfach und zeitnahe festzustellen.



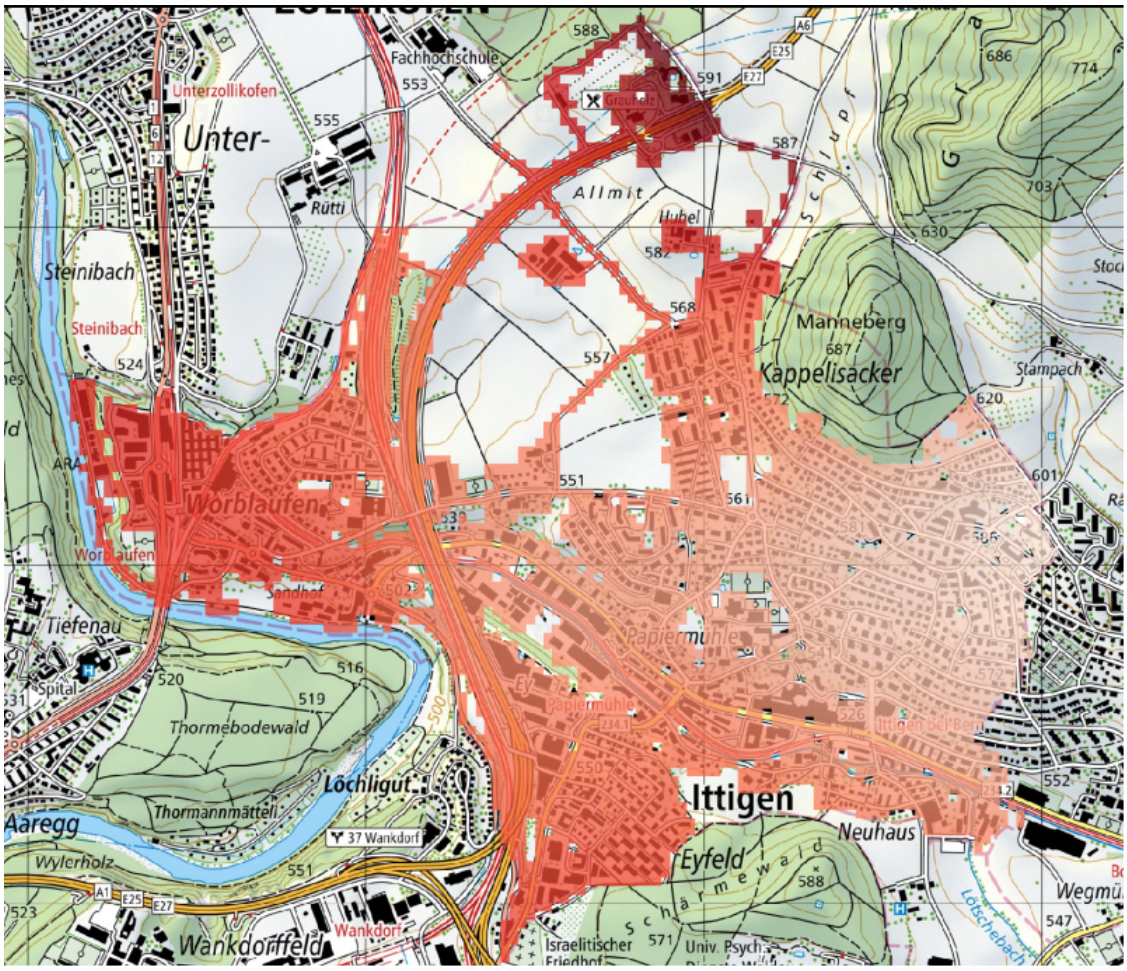
Berechnung definiert in: https://www.haupt.ch/Verlag/Buecher/Natur/Umwelt-Oekologie/Zersiedelung-messen-und-begrenzen.html

Abbildung 0.1. Schritte der definierten Berechnung

[6] https://www.are.admin.ch/are/de/home.html

Ziele, Vorgehen, Technologien

Als Werkzeug für die Berechnung dieses Indikators dient das Desktop-Geoinformationssystem QGIS. QGIS ist in C++, Qt und Python geschrieben und für alle Betriebssysteme als Open Source verfügbar. Das Ziel ist es, eine entsprechende Erweiterung in Python zu schreiben, die ebenfalls Open Source sein soll. Nach einer Evaluation fiel der Architekturentscheid auf sogenannte Processing-Algorithmen als flexibelste Erweiterungen. Im Gegensatz zu QGIS-Plugins kann man Processing-Algorithmen nach dem Baukasten-Prinzip zu 'Modellen' zusammenfügen. Somit kann die gesamte Berechnung der Zersiedlung in einzelne Processing-Algorithmen (Rechenschritte) aufgeteilt werden (vgl. Abb. 0.1). Dies erlaubt es den Benutzern, Zwischenresultate zu überprüfen (vgl. Abb. 0.2). Und es ermöglicht auch die Berechnung ohne Programmierung anzupassen und künftig zu verbessern.



Geodaten ARE, Basiskarte map.geo.admin.ch

ABBILDUNG 0.2. Dispersion kartographisch dargestellt: dunkelrote Gebiete zeigen grosse Zerstreuung innerhalb des Untersuchungsgebiets

Damit dies erreicht werden kann, muss der definierte Algorithmus zuerst analysiert und verstanden werden. Danach müssen die einzelnen Rechenschritte nacheinander implementiert werden.

Ergebnisse

Als Resultat der Arbeit wurden als einzelne QGIS-Processing-Algorithmen entwickelt: siehe Sprawl at index (Si), Dispersion (DIS), Flächeninanspruchnahme (FI) und Zersiedelung (Z) in Abb. 0.1. Diese wurden zusätzlich zu einem einzigen Processing-Algorithmus - dem 'Urban Sprawl Calculator' - zusammengefügt, um den Benutzern den Einstieg zu erleichtern (vgl. Abb. 0.3). Das Set von Processing Algorithmus hat den Namen 'Urban Sprawl (USL)' bekommen. Zusammen mit der benutzerfreundlichen Lösung wurde damit das Hauptziel vollständig erreicht. Im letzten Sprint wurde noch versucht, die zeitkritischen Teile des Algorithmus durch Plattform-übergreifende Parallelisierung mit Hilfe der vorgegebenen PyQt-Klassen zu beschleunigen, was jedoch zu keinen befriedigenden Resultaten führte. Die Berechnung einer mittelgrossen Gemeinde in der Schweiz ist trotzdem akzeptabel. In jedem Falle kann damit dem ARE ein nützliches Werkzeug übergeben werden bereit für den Einsatz in der Planung. Die Erfahrungen, die im Rahmen dieses Projektes gemacht wurden, wurden in der Dokumentation, für die Weiterentwicklung, festgehalten.

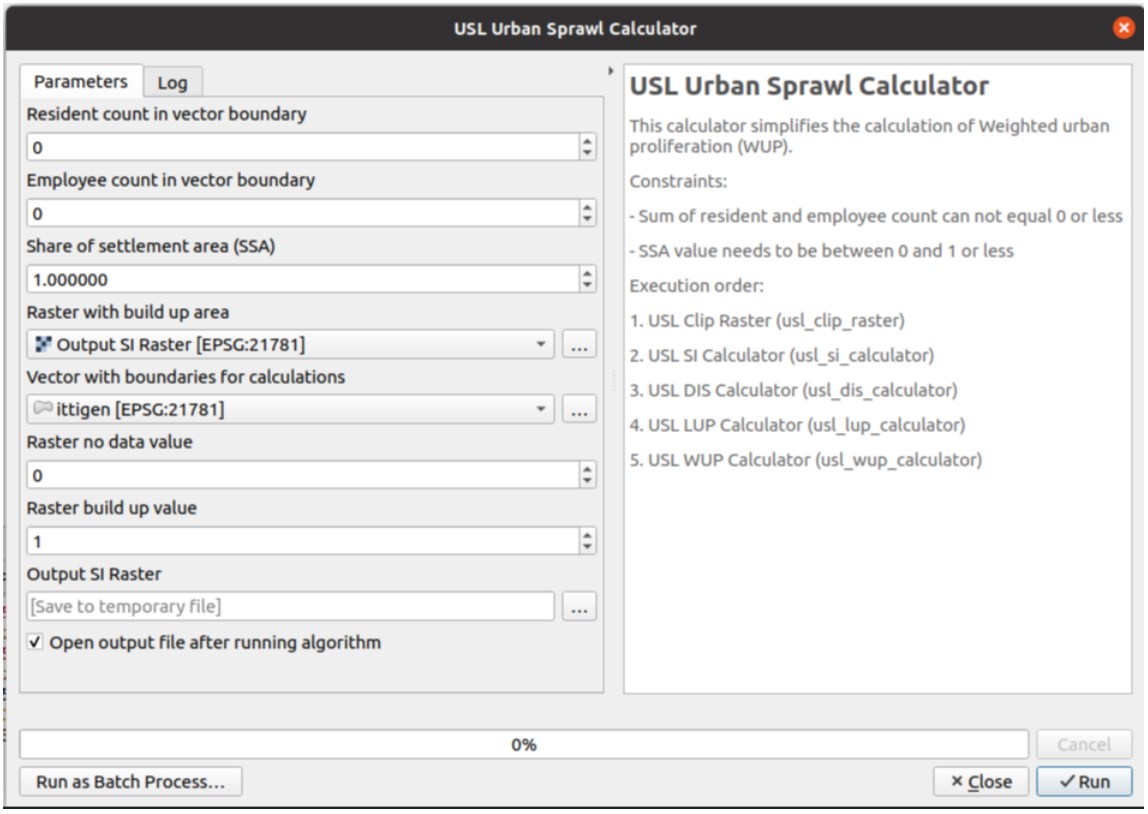


Abbildung 0.3. Dialog in QGIS zur vollständigen Berechnung des Zersiedelungs-Indikators

Ausblick

Es gibt mehrere Möglichkeiten, auf die sich die Urban Sprawl QGIS Erweiterung weiterentwickeln lässt. Einer der wichtigsten Schritte ist die Implementation der Parallelisierung bei der Berechnung. Dies würde die Rechenzeit drastisch verkürzen und dem Benutzer ein angenehmeres Erlebnis bieten.

Die nächste mögliche Weiterentwicklung wäre das Konfigurieren einfacher zu gestalten, damit das Tool für breitere Zwecke nutzbar ist. Ein wichtiger Aspekt, den man konfigurierbar machen sollte, ist die Gewichtung. Wenn die Gewichtung einfach zu konfigurieren ist, ist es auch möglich, andere Radien zu nutzen.

Die letzte wichtige Änderung ist die Möglichkeit, mehrere Berechnungen am Stück durchzuführen und diese Ergebnisse zwischen zu speichern oder sogar grafisch darzustellen in Form eines Diagrammes.

Inhaltsverzeichnis

Teil I.

Technischer Bericht

Einführung 1

1.1. Problemstellung

Die Zersiedelung ist das ungestüme Siedlungswachstums. Das Abbild dieses Wachstums ist oft in von dem Ortskern entfernten Gebieten ersichtlich. Die Sektion Grundlagen des Bundesamtes für Raumentwicklung [1] (ARE) will prüfen, wie der Indikator in der Raumplanung eingesetzt werden kann. Als Vorlagen für dieses Projekt gibt es die Literatur 'Zersiedelung messen und begrenzen' [2]. Als weitere Hilfe gab es die Dokumentation und Testdaten des USM-Toolsets [3] entwickelt von der WSL [4]. Dr. rer. nat. Hans-Georg Schwarz-von Raumer [5] stellte sein C++ Skript für die Berechnung der Dispersion als Vorlage zu Verfügung.

Als Zielpublikum dieser Arbeit sind einfache Raumplanerbüros.

1.2. Vision

Die Zersiedelung kann durch Planende gemessen werden – als Kern dient hierfür QGIS. Das Instrumentarium umfasst eine Werkzeugkiste, die es den Planenden ermöglicht einfach und zeitnah den Effekt ihrer planerischen Intervention auf die Zersiedelung zu messen. Im Wesentlichen wird das Siedlungsgebiet als Fläche 'aufgezeichnet', die Anzahl Beschäftigte und Erwerbstätige attribuiert und die Berechnung ausgelöst. Die Resultate zeigen den Effekt auf, wie die Zersiedelung sich entwickelt.

1.3. Ziele

Das Ziel dieser Arbeit ist es, entweder ein QGIS Plugin oder mit dem Processing Framework von QGIS die Berechnung der Zersiedelung zu Implementieren. Die Messgrösse Zersiedelung (Z) setzt sich aus zwei weiteren Berechnungen zusammen: die Streuung der Siedlungsflächen und die Flächeninanspruchsnahme.

Das Ziel dieser Arbeit ist es zu beweisen, dass die Berechnung der Zersiedelung einfach ohne hohe Kosten gemacht werden kann und somit für eine breitere Masse zu Verfügung steht.

1.4. Performance

Es handelt sich in dieser Arbeit um die Implementation mehrerer grossen Berechnungen. Die Performance dieser Berechnung hat bei dieser Arbeit jedoch nicht die höchste Priorität. Die Berechnung sollte jedoch in absehbarer Zeit durchgeführt werden.

1.5. Erweiterung der Funktionalität

Damit man die Zersiedelung Z Berechnen kann, werden mehrere Algorithmen gebraucht. Die Definition dieser Algorithmen kann sich ändern oder die Berechnung kann mit neuen Algorithmen erweitert werden. Aus diesem Grund sollte es möglich sein, das Processing Plugin einfach abzuändern oder zu erweitern.

1.6. Rahmenbedingungen

Für diese Arbeit werden mehrere Rahmenbedingungen festgelegt.

Sprache der Dokumentation	Deutsch
Sprache der Software	Englisch
Technologien	Pyhton 3, QGIS, PostgreSQL

Die nicht funktionalen Anforderungen werden in Kapitel 5 genauer beschrieben.

1.7. Vorgehen

Als Vorgehensmodell wurde das Rational Unified Process (RUP) gewählt. Eine genauere Beschreibung dazu ist in Kapitel 11 beschrieben. Für die Softwareentwicklung wird der Prozess Git Feature Branch Workflow [6] verwendet. Das ganze Projekt wird in Zusammenarbeit mit dem Bundesamt für Raumentwicklung ARE gemacht.

Stand der Technik 2

2.1. Schwick und Jäger

Die Zersiedelung zu berechnen ist kein neues Thema in der Raumplanung. Damit man diese überhaupt berechnen kann, wird ein Algorithmus benötigt. Dieser Algorithmus wurde von Schwick und Jäger [2] beschrieben und definiert. Alle bestehenden Produkte, mit denen man die Zersiedelung berechnet, haben diesen Algorithmus implementiert.

2.2. WSL USM-Toolset

Die Eidgenössische Forschungsanstalt für Wald, Schnee und Landschaft (kurz WSL)[4] hat diesen Algorithmus bereits in einem Projekt implementiert. Das dabei entstandene Tool nennt sich USM-Toolset. Dieses ist eine Erweiterung für das Geo Informationssystem (GIS) ArcGIS [7]. Damit dieses Tool in ArcGIS benutzt werden kann, werden weitere Erweiterungen und eine spezielle Konfiguration gebraucht. Aus diesen Gründen ist die Benutzung des USM-Toolset sehr teuer. Dies hat zu Folge, dass es nicht allen Firmen zu Verfügung steht.

Umsetzungskonzept 3

3.1. Beschreibung

Durch die Aufgabenstellung wurde bereits klar, dass das Projekt eine Erweiterung für QGIS sein wird, mit der die Berechnung einfach durchfürhbar ist.

Für die Umsetzung einer solchen Erweiterung gibt es drei Möglichkeiten. Die erste Möglichkeit ist es, das Ganze als Plugin zu entwickeln, welche die Endnutzer schlussendlich über den Plugin Manager von QGIS installieren können. Die zweite Möglichkeit wäre es, das Ganze als Processing Algorithmus zu implementieren. Processing Algorithmen sind kleine Skripts, die man auch aneinanderketten kann, um schlussendlich das Resultat zu erreichen. Bei den Processing Algorithmen ist es möglich diese auch über den Plugin Manager zu installieren. Die letzte Möglichkeit ist es die ganze Rechnung auf einem externen Server auszuführen. Es gibt mehrere Möglichkeiten, wie QGIS eine Anfrage an das Backend sendet.

Die Entscheidung ist schlussendlich auf die Processing Algorithmen gefallen, da diese den Vorteil haben, dass alle Untergrössen von der Zersiedelung als eigenes Skript implementiert werden können. Dies erlaubt das einfache Zwischenspeichern von allen Resultaten und falls notwendig auch einfacheres Abändern der Berechnung.

Genauere Beschreibungen der Entscheidungen können in der Software Dokumentation in Kapitel 9 gefunden werden.

Resultate 4

4.1. Zielerreichung

Mit den Urban Sprawl Processing Algorithmen lässt sich sehr einfach die Zersiedelung in einem bestimmten Bereich berechnen. Da das ganze Tool und die benötigte Software Open Source ist, sind diese Processing Algorithmen für jeden frei verfügbar. Dies gibt jedem Geo Informatiker die Option, dieses Projekt weiterzuentwickeln oder für ihren Anwendungsfall anzupassen.

Neben der Berechnung von Z lässt sich auch jeder Teil der Berechnung einzeln ausführen. Der Vorteil davon ist es, dass der Benutzer bestimmte Parameter verändern und einen bestimmten Schritt wiederholen kann.

Der Git Verlauf des Projektes wird zurückgesetzt und alle Code Änderungen mit einem Commit abgebildet. Der Grund dafür ist, dass viele Prototypen, die unteranderem fehlgeschlagen sind, nicht Öffentlich sein sollen.

4.2. Ausblick: Weiterentwicklung

4.2.1. QGIS Plugin

Momentan erfolgt die Installation der QGIS Processing Algorithmen manuell. Eine Weiterentwicklung wäre über den QGIS Pluginstore zu machen. Dies bedeutet, dass der User nur noch das Urban Sprawl Plugin auswählen muss, welches dann die Processing Algorithmen zu QGIS hinzufügt.

4.2.2. Parallelisierung

Wie beschrieben in Unterunterabschnitt 9.2.8.1 wurde keine Parallelisierung angewandt. Eine Erweiterung wäre es die Berechnung von SI zu parallelisieren, was die Berechnung um ein Vielfaches beschleunigen würde. Obwohl die Berechnung nicht parallelisiert ist, lässt sich der SI Wert einer mittelgrossen Gemeinde in der Schweiz in einer akzeptabler Zeit berechnen.

4.2.3. Mehrere Berechnungen

Es gibt momentan nicht die Möglichkeit einfach mehrere Z Berechnungen an einem Stück durchzuführen. Ein Ziel in der Zukunft ist es, dass der Benutzer mehrere Berechnungen am Stück durchführen kann und diese Ergebnisse zwischengespeichert werden, sodass am Ende das Resultat grafisch dargestellt werden kann. So kann der Benutzer zum Beispiel die Entwicklung der Zersiedelung eines Gebietes anschauen.

Ausserdem kann man auf diesem Weg auch den Unterschied verschiedener Interventionen eines Raumplaners grafisch darstellen.

Teil II.

Software-Projektdokumentation

Anforderungsspezifikation 5

5.1. Nicht Funktionale Anforderungen

NFR-01 Wartbarkeit: Die Wartbarkeit am System soll so einfach wie möglich gehalten sein. Dies wird unterstützt mit Hilfe der Dokumentation.

NFR-02 Open Source: Das ganze Projekt soll Open Source sein und für jede Person sowohl einsehbar, als auch abänderbar sein.

NFR-03 Gebrauchstauglichkeit: Die Beschreibung der Processing Algorithmen sowie der Text der Inputdaten soll für alle verständlich sein. Falls dies nicht der Fall ist, gibt es eine Benutzeranleitung, die dabei Abhilfe schafft.

NFR-04 Installierbarkeit: Die Erweiterung soll einfach in QGIS installierbar sein. Damit dies erfüllt ist, wird sichergestellt, dass man das Tool mit dem Plugin Manager installiert werden kann.

5.2. Personas

Name	Erfahrungen	Ziele
Hans Jürgens	Hat persönliches Interesse an Geografie und die Entwicklung der Schweiz. Er hat schon mehrfach mit QGIS gearbeitet und kennt dadurch die Grundfunktionalitäten davon.	Er will die Zersiedelung von seiner Wohngemeinde für mehrere Jahre berechnen, um den Unterschied festzustellen. Das Gebiet befindet sich ausschliesslich in der Schweiz.
Sonya Mellar	Hat Geografie und Informatik studierte und in beiden Bereichen mehrjährige Berufserfahrung. Sie hat schon mehrfach mit QGIS gearbeitet und auch schon eigene Plugins und Processing Algorithmen geschrieben.	Sie will die Zersiedelung mehrerer Gebiete in der Schweiz berechnen. Diese sind nicht ausschliesslich Gemeinden, sondern auch kleinere Orte.

Tabelle 5.1. Personas

5.3. Szenarien

5.3.1. Szenario 1: Hans Jürgens

Hans Jürgens will von seiner Wohngemeinde die Zersiedelung berechnen. Er hat QGIS mit der Urban Sprawl Erweiterung bereits installiert. Die Daten seiner Wohngemeinde besorgt er sich mithilfe des Bundesamts für Statistik. Ihn persönlich interessiert nur das Endresultat, um eine

Statistik zu erstellen. Dazu rechnet er mit dem USL Processing Algorithmus alles an einem Stück. Das Resultat der Berechnung speichert er sich in einer Exceltabelle. Er berechnet einen Zersiedelungswert nach dem anderen.

5.3.2. Szenario 2: Sonya Mellar

Sonya Mellar hat den Wunsch, die Zersiedelung von kleineren Gebieten genauer anzuschauen. Sie hat QGIS bereits auf ihrem Computer installiert und das Zip File für die Urban Sprawl Erweiterung heruntergeladen. Da die Gebiete jedoch klein sind, will sie noch den Radius und die Gewichtung anpassen. Im Code der Urban SprawlErweiterung ändert sie die Gewichtung so ab, dass es zu ihrem gewünschten Radius stimmt. Den Radius passt sie nicht direkt im Code an. In QGIS erstellt sie sich selber ein Model mit dem Modellbaukasten, indem sie die benötigten USL Processing Algorithmen verbindet. Den Radius wählt sie dabei als möglichen Input. Dieses Model speichert sie, damit es später noch verwendet werden kann.

Analyse 6

6.1. Domainmodell

Da die Berechnung als separate Processing Algorithmen implementiert werden, entsteht kein Domainmodell. Vielmehr handelt es sich um eine Analyse des Algorithmus für die Berechnung der Zersiedelung Z.

6.2. Berechnung

Wesentlich ist die Berechnung der Zersiedelung Z. Dieser Algorithmus lässt sich in mehrere kleine Algorithmen unterteilen.

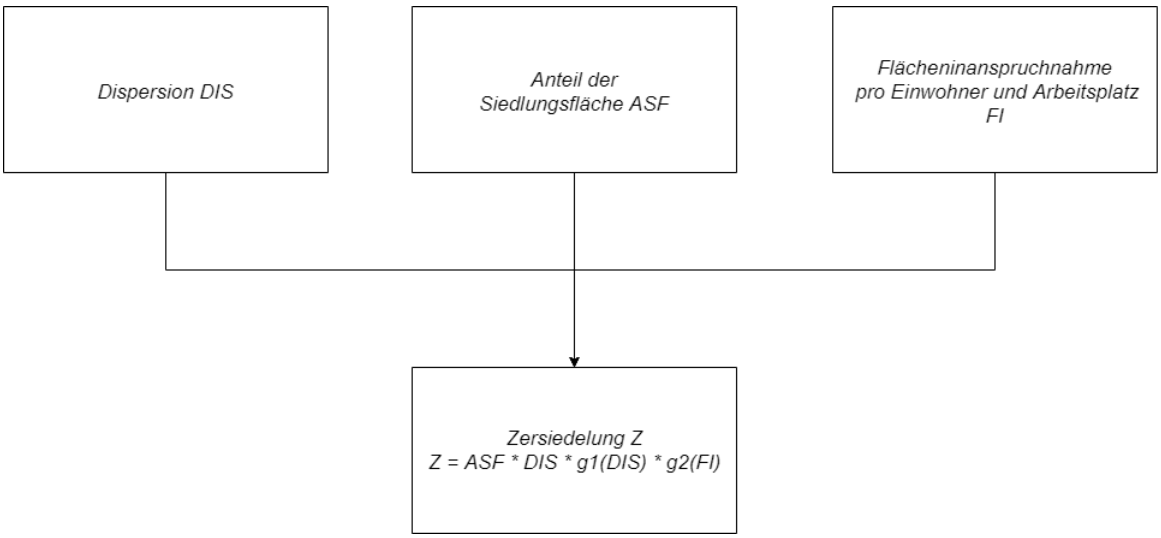


Abbildung 6.1. Berechnung der Zersiedelung, wie beschrieben in Schwick und Jäger [2]

DIS: Die Dispersion (DIS) ist die Streuung der Siedlungsfläche in einem Gebiet. Dazu wird zuerst der SI Wert berechnet. Für den SI Wert wird ein Siedlungspixel im Raster genommen und davon der Durchschnitt der Distanzen zu allen anderen Siedlungspixel im Raster ausgerechnet. Diese Berechnung wird für alle Siedlungspixel im Raster durchgeführt. Von den Resultaten wird der Durchschnitt berechnet, welches der DIS Wert ist.

FI: Die Flächeninanspruchnahme (FI) sagt aus, wie viele Arbeitende oder Einwohner es pro Quadratmeter in dem bebauten Gebiet gibt.

Anteil der Siedlungsfläche (ASF): Der Anteil der Siedlungsfläche (ASF) sagt aus, wieviel der Siedlungsfläche in Betracht gezogen werden muss. Dieser Wert wird benutzt, damit Orte, die zu einem grossen Teil aus einem See oder Bergen bestehen, nicht den Wert der Zersiedelung verfälschen.

Z: Die Zersiedelung (Z) ist das Endprodukt dieser Berechnung. Diese Grösse besteht aus dem gewichteten DIS Wert, dem gewichteten FI Wert und dem ASF Wert, der noch mit der

Dispersion multipliziert wird.

Wichtig zu beachten ist auch noch die Gewichtung der Algorithmen. Diese sind mathematische Funktionen, die auf die jeweiligen Werte noch angewendet werden. Die Gewichtung wird in Unterabschnitt 9.2.6 genauer beschrieben.

Sicherheit 7

Diese Arbeit hat keine kritische Probleme bezüglich der Sicherheit, da alles auf dem lokalen Rechner läuft und keine externen Ressourcen verwendet werden. Alle Python Bibliotheken, die verwendet werden, kommen von QGIS.

Ein Aspekt, den man aber in Betracht ziehen muss, sind die Inputs, die man für die Processing Aglorithmen verwendet. Für die Berechnung Z wie auch FI braucht man das Verhältnis zwischen Wohnhaften und Arbeitenden in einem bestimmten Gebiet, diese Daten werden aber vom User selbst verwendet, daher liegt die Verantwortung beim Anwender der Software.

Design 8

8.1. Architektur

Der Urban Sprawl Calculator (USL) ist ein Processing Algorithmus, welcher wiederum andere Processing Algorithmen aufruft. Der Hauptprocessing Algorithmus ist eigentlich nur ein Wrapper, welcher das Aufrufen der einzelnen Processing Algorithmen vereinfacht. Dies wurde so implementiert, damit der Benutzer die Möglichkeit hat, die einzelne Schritte manuel auszuführen, und einzelne Werte zu verändern. So ist es möglich, dass der Benutzer das SI Raster zwischenspeichert, da diese Berechnung am längsten dauert, und er die Eingabewerte der restlichen Algorithmen abändern kann.

Um die Implementation zu vereinfachen, verwenden die Processing Algorithmen ein **Common** Package. Dieses beinhaltet Code, der von mehreren Processing Algorithmen verwendet werden. Dies hat der Vorteil, dass es keinen duplizierten Code gibt. Ausserdem wird damit die Komplexität der einzelnen Processing Algorithmen verringert.

8.2. Deployment

Das Deployment wurde so simpel wie möglich gehalten. Der Benutzer muss dazu nur ein Zip File herunterladen und über den Plugin Manager installieren.

8.3. Paketdiagramm

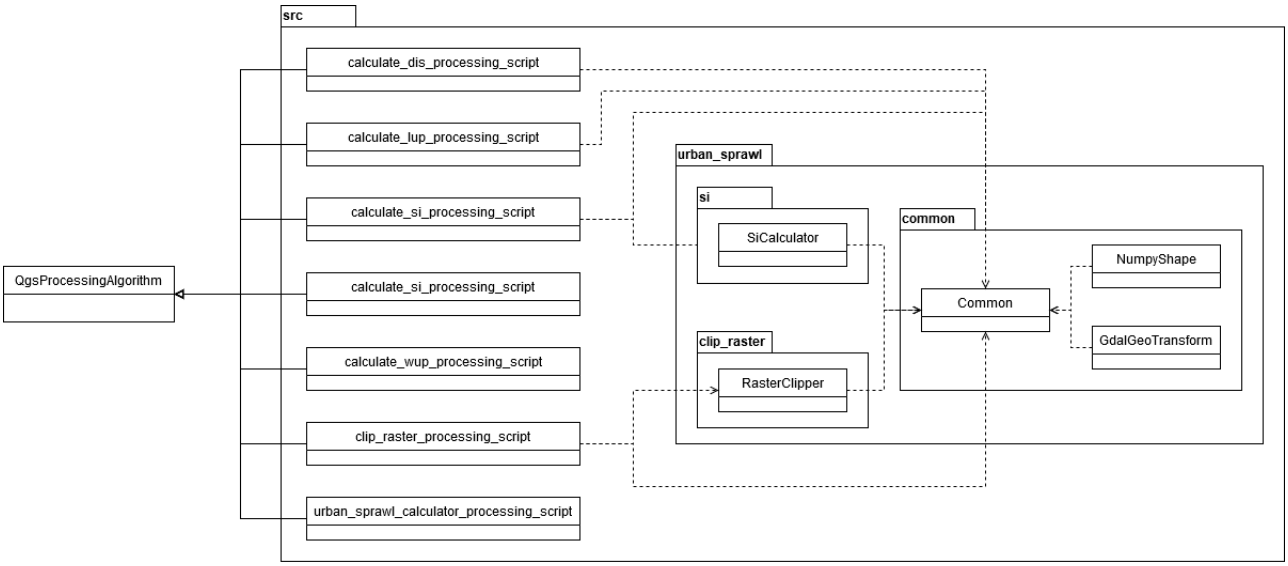


Abbildung 8.1. Paketdiagramm

Der ganze Code befindet sich im Modul **src/**. Im Root dieses Ordners befinden sich alle Processing Algorithmen.

Im Ordner **src/urban_sprawl/** befindet sich gemeinsamer Code, wie auch Klassen, die von einzelnen Processing Algorithmen verwendet werden. Dieser wurde nicht als externe Library implementiert, damit man bei der installation kein Pip oder andere Tools verwenden muss.

8.4. Sequenzdiagramme

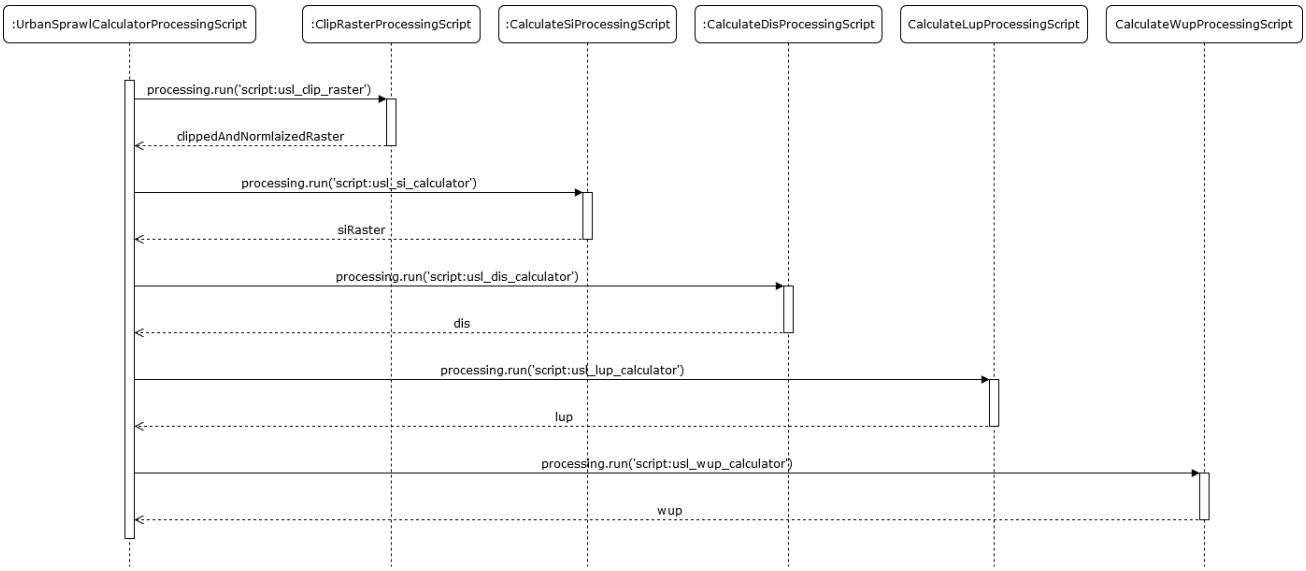


ABBILDUNG 8.2. Sequenzdiagramm

In diesem Sequenzdiagramm sieht man, wie der Hauptprocessing Algorithmus (UrbanSprawl-CalculatorProcessingScript) die verschiedenen Algorithmen sequenziell aufruft und am Schluss das Resultat **WUP** bekommt.

8.5. User Interface

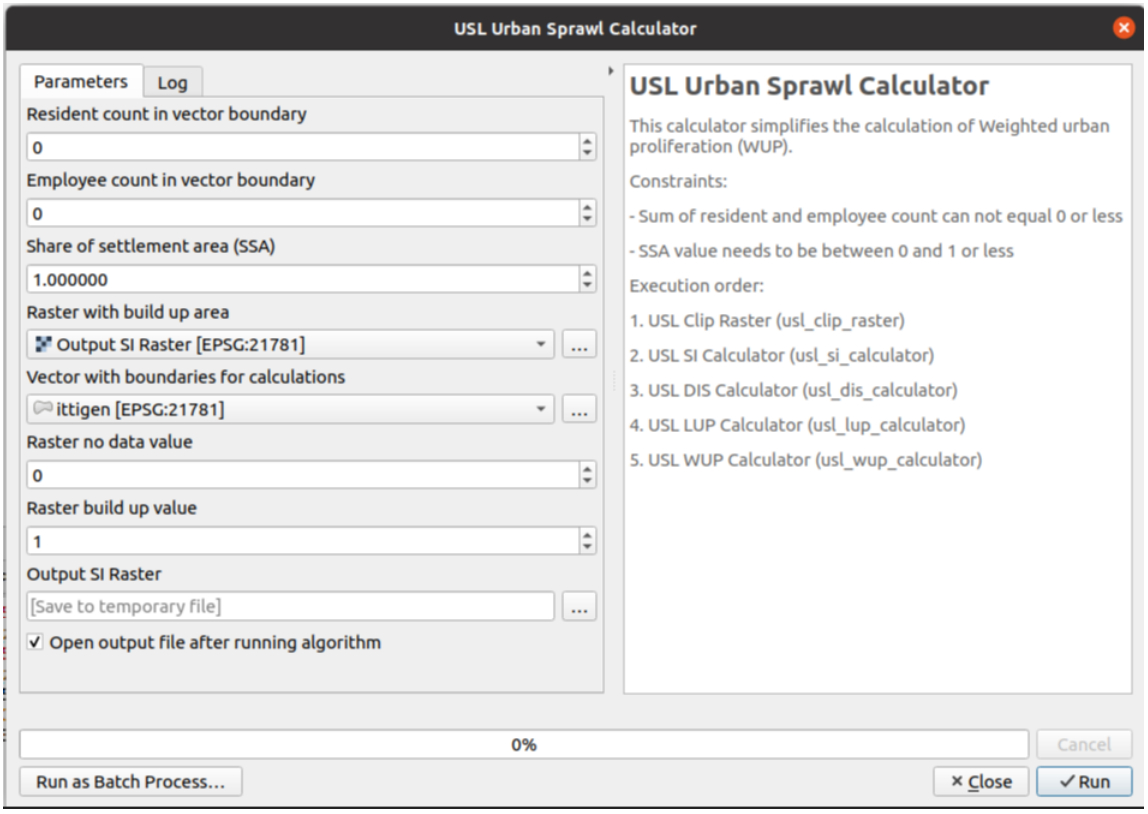


ABBILDUNG 8.3. User interface für Z Berechnung

Über das User Interface hat man sehr wenig Kontrolle, wenn man ein Processing Algorithmus verwendet. Das Einzige, das man setzen kann, sind die verschiedenen Input- und auch gewisse Outputfelder. Des weiteren kann noch ein Titel und eine Beschreibung gesetzt werden, jedoch hat man keine Kontrolle über die Formatierung oder Darstellung.

9 Implementation

9.1. QGIS Erweiterung

Wenn man eine Erweiterung für QGIS schreibt, dann gibt es zwei direkte und eine indirekte Möglichkeit. Die beiden direkten sind ein Plugin oder Processing Algorithmen zu erstellen. Die indirekte ist ein Backendserver, der die Berechnung durchführt. Alle diese Möglichkeiten haben verschiedene Vor- und Nachteile, die in diesem Abschnitt genauer erläutert werden.

9.1.1. Plugin

Eine der direkten Möglichkeiten ist es, dass man als Erweiterung ein Plugin macht. Das Plugin lässt sich sehr einfach über den QGIS Plugin Manager installieren und verbreiten. Beim Erstellen eines Plugins kann das GUI sehr einfach spezifiziert werden. Dem Entwickler ist dabei die Freiheit überlassen.

Jedoch ist das Ganze eine Blackbox, wo alles einfach fix an einem Stück durchläuft. Die Plugins stehen immer alleine und können nicht von anderen Plugins abhängig sein.

9.1.2. Processing Algorithmus

Processing Algorithmen sind eine andere Art einer direkten QGIS Erweiterung. Ein Processing Algorithmus ist dafür da, genau eine Funktionalität zu haben oder eine Berechnung durchzuführen. Es ist ausserdem möglich, mehrere dieser Processing Algorithmen mithilfe des Modellbaukastens aneinanderzureihen, damit diese am Stück ausgeführt werden. Da alles von einander getrennt implementiert wird, ist es auch sehr einfach, alle Schritte separat zu den anderen laufen zu lassen und auch gewisse Teile auszutauschen.

Installieren kann man das Ganze zwar über den Plugin Store, jedoch muss dafür ein Plugin geschrieben werden, das die Installation übernimmt.

9.1.3. Backendserver

Die Implementation könnte man auch indirekt machen, indem man alle Berechnungen auf einem Backendserver ausführt. Dies würde alle Abhängigkeiten von QGIS entfernen, da QGIS bei dieser Möglichkeit nur als Frontend gilt. So wäre es auch möglich, noch weitere Frontends an diesen Server anzuschliessen.

Da die Berechnung sehr aufwendig ist und ein Backend von mehreren Benutzern gleichzeitig benutzt werden kann, muss es sich um einen leistungsstarker Server handeln, was sehr teuer wird. Ausserdem muss dieser Server auch noch deployed werden, was zu weiteren Problemen und Risiken führen kann.

9.1.4. Entscheidung

Der Backendserver macht bei dieser Arbeit keinen Sinn, da nur Kosten und Risiken dazu kommen. Es ist auch nicht nötig, dass es weitere Frontends gibt für dieses Projekts. Es wäre nur nötig diese Möglichkeit zu nutzen, wenn es mit QGIS unmöglich ist, den Algorithmus zu implementieren.

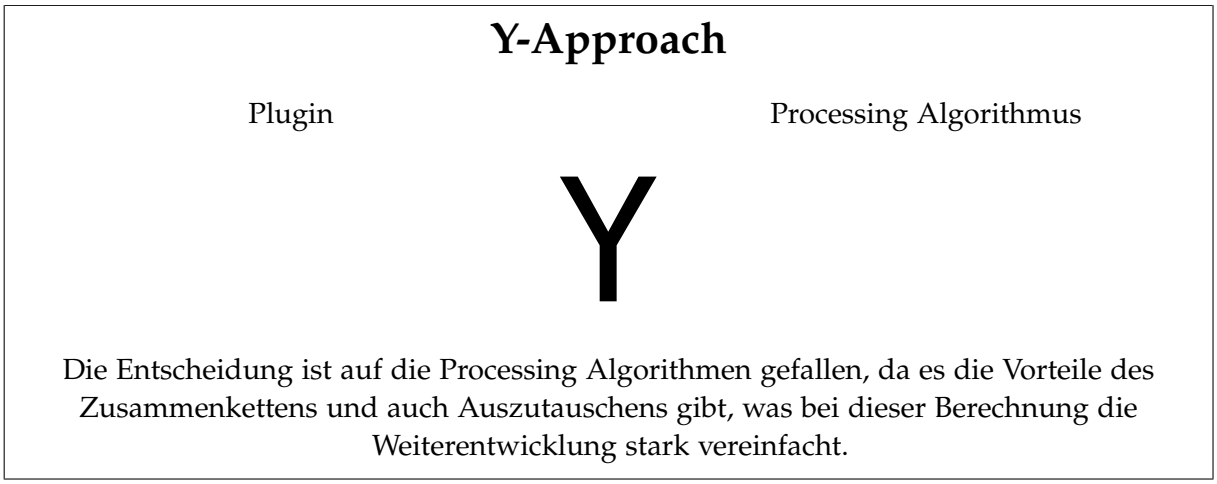


Abbildung 9.1. Y-Approach [8]

9.2. Berechnung

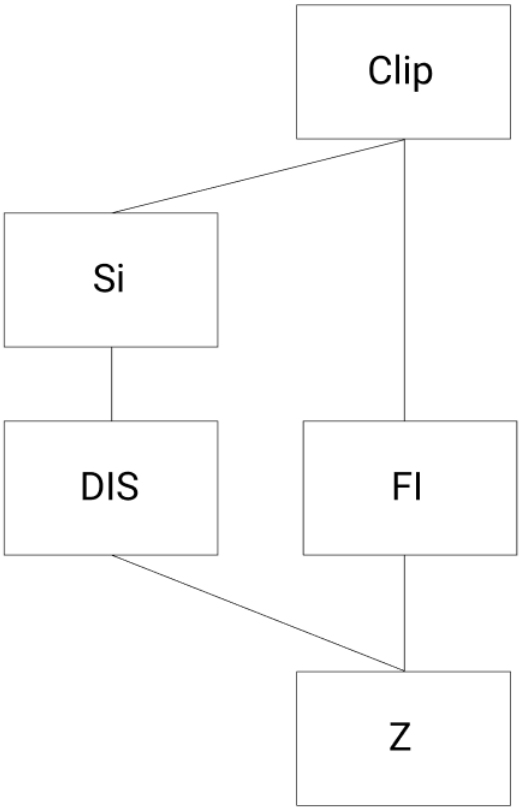


Abbildung 9.2. Berechnung der Zersiedelung

Die Berechnung der Zersiedelung wurde in mehrere Processing Algorithmen verteilt. Dies hat den Vorteil, dass alles entweder an einem Stück oder einzeln durchgerechnet werden kann, je nach dem Bedürfnis des Benutzers.

9.2.1. Input Daten

Die Inputdaten, wenn man alles an einem Stück durchrechnet, sind so einfach wie möglich gehalten, so dass es jeder auch ohne grosse Vorkenntnisse bedienen kann. Die Inputdaten sind:

- Raster mit dem bebauten Gebiet
- Ein Vektor Polygon mit den Gebietsgrenzen
- Anzahl der Bewohner in dem Gebiet
- Anzahl der Angestellten in dem Gebiet
- Der ASF Wert für die Gewichtung

Die Anzahl der Bewohner oder der Angestellten kann auch zusammen eingetragen werden. Dies erreicht der Benutzer einfach, indem er einen der Werten auf 0 lässt.

Der Radius ist konstant auf 2'000 Meter gesetzt, da je nach Radius die Gewichtungsfunktionen anders sind. Mehr dazu in Unterabschnitt 9.2.6.

9.2.2. Clip Raster Layer

Der Clip Raster Layer Processing Algorithmus ist der Erste, der aufgerufen werden sollte. Dieser stellt sicher, dass nur die Fläche in den Gebietsgrenzen genutzt wird. Dies wird sichergestellt, indem wir die Daten des Rasters normalisieren.

Für das Ausschneiden des Rasters aus dem Polygon wurde von GDAL [9] ein Processing Algorithmus verwendet. Der Algorithmus von GDAL mit dem Namen 'Clip raster by Mask' braucht als Input ein Vektor Polygon und ein Raster und gibt als Output das ausgeschnittene Raster zurück. Dieses Raster wird dann in eine Numpy [10] Matrix übertragen und schlussendlich normalisiert. Diese Matrix wird danach wieder in ein Raster umgewandelt, welches der Output des Algorithmus ist.

9.2.3. SI Berechnung

Die Berechnung von SI [11] ist eine der wichtigsten Berechnungen, da diese am längsten dauert. Bei der Berechnung von SI handelt es sich darum, dass ein Pixel im Raster genommen wird, der als bebaut markiert wurde. Von diesem Pixel aus wird die Distanz zu allen anderen Pixel, die als bebaut markiert und in einem bestimmten Radius sind, aufsummiert. Davon wird der Durchschnitt berechnet. Dies macht man für jeden bebauten Pixel im Raster.

Es werden zwei Raster gebraucht. Eines welches vom ersten Processing Algorithmus (Clip Raster Layer) erzeugt wird (Clipped Raster) und ein zweites, welches über die Gebietsgrenzen hinaus geht. Das zweite Raster wird benötigt, damit die Berechnung der Dispersion später genauer ist.

Die beiden Raster werden in Numpy Matrizen umgewandelt. Die Clipped Raster Matrix wird mit einem Loop durchlaufen. Wenn ein Pixel als bebaut markiert ist, wird die Position dieses Pixels gespeichert und im zweiten Raster gesucht. Diese Koordinaten sind der Mittelpunkt für die Berechnung. Das zweite Raster wird in zwei Loops durchlaufen. Der erste Loop in die X-Richtung und der zweite in die Y-Richtung. Die Grenzen der Loops wird der Radius geteilt durch die Pixelgrösse gesetzt. Dadurch weiss man wie viele Pixel überprüft werden müssen in der Berechnung. Dies hat den Vorteil, dass nicht bei jedem Loop überprüft werden muss, ob der Pixel sich im Radius befindet.

Für die Pixel, die sich im Radius befinden und als bebaut markiert sind, wird die euklidische Distanz [12] zum Ursprungspixel aus berechnet.

$$\sqrt{\sum_{i=1}^{2}(y_i - x_i)^2}$$

Die Distanzen werden aufsummiert und die Anzahl gezählt. Davon wird der Durchschnitt berechnet und am Ende wird dieser Wert in einer neuen Matrix, an dieselbe Position, wie der Ursprungspixel geschrieben.

Aus dieser neuen Matrix wird dann ein Raster erzeugt mit den neuen Werten. Dieses Raster kann man einfärben, um die Dispersion noch farblich darzustellen.

9.2.4. DIS Berechnung

Als Input für die DIS Berechnung wird das vorher berechnete SI Raster benötigt. Der DIS Wert sagt aus, wie gross die Dispersion im ganzen Gebiet ist. Dafür wird das SI Raster in eine Matrix umgewandelt und alle Pixel, die einen SI Wert gehabt haben, werden zusammen gezählt und der Durchschnitt daraus berechnet.

9.2.5. FI Berechnung

Für die Berechnung des FI Wertes wird sowohl die ausgeschnittene Siedlungsfläche als auch die Anzahl der Einwohner und Angestellten in der Gebietsgrenze gebraucht. Die beiden Anzahlen werden zusammen gezählt und durch die Siedlungsfläche geteilt.

9.2.6. Gewichtung

Für die Berechnung der Grösse Z müssen der DIS Wert und der FI Wert gewichtet werden. Wichtig bei der Gewichtung ist zu beachten, dass diese Formeln nur gelten, wenn der Radius auf 2'000 Meter gesetzt wird. Ansonsten müssen die Formeln auf den ausgewählten Radius angepasst werden. Die Formeln sind definiert nach Schwick und Jäger [2].

9.2.6.1. DIS Gewichtung

Die Gewichtung der Dispersion ist ein Wert, der sich zwischen 0.5 und 1.5 befindet. Das Ziel dieser Gewichtung ist es, dass höhere Gebiete mit einer höheren Dispersion einen grösseren Einfluss haben und weniger disperse einen kleineren.

Die Formel für die Gewichtung:

$$0.5 + \frac{e^{0.294432 * DIS - 12.955}}{1 + e^{0.294432 * DIS - 12.955}}$$

9.2.6.2. FI Gewichtung

Die Gewichtung des FI Wertes befindet sich zwischen 0 und 1. Wenn die Fläche besser genutzt wird, sollte diese einen kleineren Einfluss haben auf die Grösse Z, als wenn diese schlecht genutzt wird.

Die Formel für die Gewichtung:

$$\frac{e^{4.159 - 613.125/FI}}{1 + e^{4.159 - 613.125/FI}}$$

9.2.7. Z Berechnung

Die Grösse Z ist die Zersiedelung und somit das Endresultat dieses Algorithmus. Als Input wird DIS, ASF und FI benötigt. Die Formel setzt sich dann wiefolgt zusammen:

$$Z = ASF * DIS * g_1(DIS) * g_2(FI)$$

Wobei g_1 die DIS Gewichtungsfunktion ist und g_2 die FI Gewichtungsfunktion.

9.2.8. Probleme

9.2.8.1. Parallelisierung

QGIS bietet eine eigene Implementation von QT [13] QThreadPool an, welche QgsTaskManager [14] heisst. Um diesen Taskmanager zu verwenden, muss man zuerst einen QgsTask [14] implementieren, welcher entweder eine Methode oder Klasse sein kann. Die Implementierung dieser Task wird auf OpenGIS [15] beschrieben.

Da die Berechnung von SI sehr lange dauert und gut parallelisiert werden kann, macht es am meisten Sinn diese Berechnung zu parallelisieren. Die kleinste mögliche Einheit, in die die SI Kalkulation unterteilt werden kann, ist ein einzelner bebauter Pixel und die Distanz-Berechnung zwischen diesem Pixel und allen Pixel in einem bestimmten Radius. Aus diesem Grund wird für jeden bebauten Pixel des Rasters einen Task erstellt, welche dann an den Taskmanager übergeben werden. Wenn alle Tasks dem Taskmanager übergeben wurden, muss nur noch gewartet werden, bis alle Tasks beendet sind. Danach kann man aus den Resultaten ein neues Raster erzeugen.

Hier tauchen die ersten Probleme auf. Je mehr Tasks man gleichzeitig dem Taskmanager übergibt, desto wahrscheinlicher wird es, dass nicht alle Tasks gestartet und dann auch beendet werden. Um dieses Problem zu lösen, wurde ein Loop implementiert, der nur eine bestimmte Anzahl von Tasks dem Taskmanager übergibt und wartet, bis diese vollendet sind. Leider stellt dies nur sicher, dass alle Tasks starten, aber nicht dass sie auch alle beendet werden. Um zu überprüfen, ob ein Task fertig ist, gibt es eine **isActive()** Methode auf dem QgsTask. Das Problem ist aber, dass in manchen Fällen immer **True** retourniert wird, auch wenn der Task fertig ist.

QgsTask haben eine **cancel()** Methode um den Task abzubrechen. Beim Aufruf dieser Methode wird ein Signal an den Task gesendet, was dazu führt, dass die Methode **isCancled() True** retourniert. Dies funktioniert aber nicht sehr zuverlässig und hat die Konsequenz, dass Tasks existieren, die nicht beendet werden können, auch wenn die Methode mehrmals aufgerufen wird.

QgsTask hat noch eine weiter Methode **status()**. Diese Methode hat das Problem, dass in manchen Fällen der Status nie von **Running** auf **Completed** oder **Termineted** wechselt.

Ein weiteres Problem ist, dass für alle Tasks ein globaler QThreadPool verwendet wird. Welcher aber auf eine fixe Anzahl Threads limitiert ist, was dazu führt, dass man eine maximale Auslastung von 60-70% auf Windows und auf Linux von 40% erzielt. Dies kann auch nicht optimiert werden, wenn man die maximale Anzahl Threads des globalen QThreadPool erhöht oder in den QGIS Einstellung die maximale Anzahl Threads anpasst.

Mehrere Tests haben auch ergeben, dass die Berechnung schneller ist ohne QGIS Parallelisierung. Eine weitere Option um zu parallelisieren ist es Python Multiprocessing [16] zu verwenden. Dies funktioniert aber mit QGIS auf Windows nicht, da Windows Threads kein **fork** haben [17].

Aus den oben genanten Gründen ist es momentan nicht möglich, QGIS Komputation zuverlässig zu parallelisieren.

Testing 10

10.1. Probleme

Am Anfang war geplant, dass Pytest mit der CI ausgeführt werden, welches sich aber als sehr schwierig herausstellte. QGIS bietet die Option externe Pythonfiles auszuführen an, besitzt selbst jedoch nicht Pytest. Also muss man ein Skript erstellen, welches von Pytest in QGIS ausgeführt wird und dann den Output validiert.

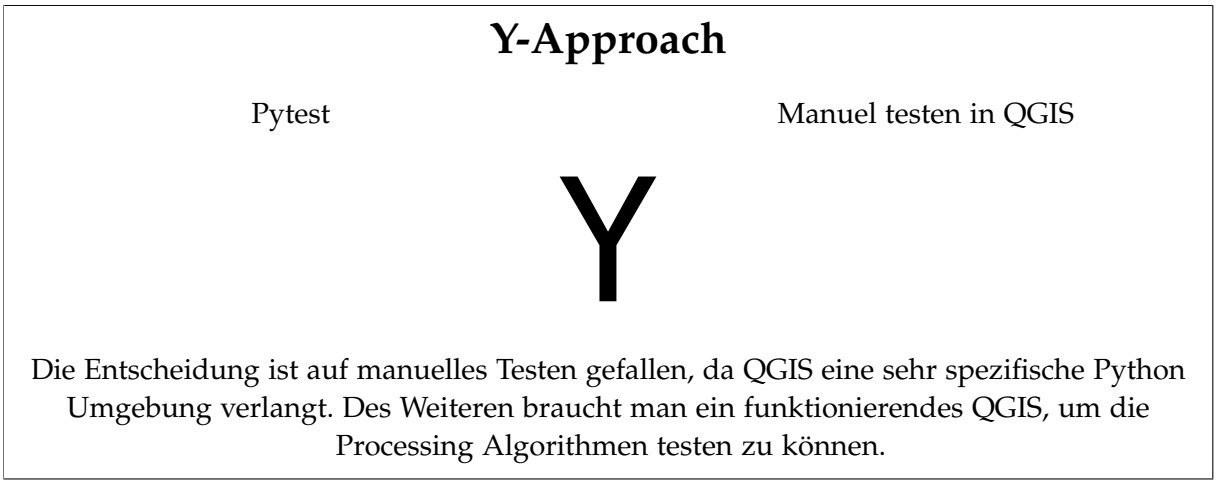


Abbildung 10.1. Y-Approach [8]

10.2. Testprotokoll

Test-Nr	Beschreibung	Bestanden	Probleme
1.1.	Import der Python Module funktioniert ohne Fehler	✓	
2.1.	DIS Processing Algorithmus returniert richtiges Resultat	✓	
2.2.	DIS Processing Algorithmus scheitert, wenn das SI Raster leer ist	✓	
3.1.	FI Processing Algorithmus retourniert richtiges Resultat	✓	
3.2.	FI Processing Algorithmus funktioniert, wenn nur **Einwohner** einen Wert grösser als 0 hat	✓	
3.3.	FI Processing Algorithmus funktioniert, wenn nur **Angestellte** einen Wert grösser als 0 hat	✓	
3.4.	FI Processing Algorithmus scheitert, wenn **Einwohner** und **Angestellte** beide 0 sind	✓	
4.1.	SI Processing Algorithmus retourniert richtiges Resultat	✓	
5.1.	Zersiedelung Processing Algorithmus retourniert richtiges Resultat	✓	
5.2.	Zersiedelung Processing Algorithmus scheitert, wenn der SSA Wert kleiner oder gleich 0 ist	✓	
5.3.	Zersiedelung Processing Algorithmus scheitert, wenn der SSA Wert grösser als 1 ist	✓	
6.1.	Clip Processing Algorithmus retourniert richtiges Resultat	✓	
7.1.	Zersiedelung Processing Algorithmus retourniert richtiges Resultat	✓	
7.2.	Zersiedelung Processing Algorithmus retourniert gleiches Resultat, wie wenn man alle Schritte manuel laufen lässt	✓	
7.3.	Zersiedelung Processing Algorithmus scheitert, wenn **Einwohner** und **Angestellte** beide 0 sind	✓	
7.4.	Zersiedelung Processing Algorithmus scheitert, wenn SSA Wert kleiner oder gleich 0 ist	✓	
7.5.	Zersiedelung Processing Algorithmus scheitert, wenn SSA Wert grösser als 1 ist	✓	

TABELLE 10.1. Testprotokoll

Projektmanagement 11

11.1. Vorgehen

Für das Projektmanagement fiel die Wahl auf das Rational Unified Process (RUP), welches aus vier Phasen besteht. Dies sind Inception, Elaboration, Construction und Transition. Jedoch wird neu noch eine weitere Phase hinten angeschlossen. Bei dieser Phase handelt es sich darum, die Dokumentation abzuschliessen. In RUP wird in den Phasen in Sprints gearbeitet. Vor jedem Sprint wird geplant, was im nächsten Sprint gemacht wird mit einer Aufwandschätzung. Ausserdem wird geplant, wie lange so ein Sprint geht, da keine fixe Sprintdauer gesetzt wurde.

11.2. Phasenplanung

Kalenderwoche	Phase	Beschreibung
8	Inception	Verständnis der Aufgabenstellung
9-12	Elaboration	Einarbeiten in die Materie
13-20	Construction	Umsetzung des Produktes
21-22	Transition	Code Freeze
23-24	Dokumentation	Fertigstellen der Dokumentation

Tabelle 11.1. Definierte Phasen

11.3. Meilensteine

Meilenstein	Datum	Beschreibung
M1: Berechnung verstanden haben	05.03.2020	Die Berechnung der Dispersion/ Streuung der Siedlungsfläche verstanden haben
M2: Prototyp	26.03.2020	Standalone Prototyp zu der Berechnung der Zersiedelung
M3: DIS Berechnung	16.04.2020	Die DIS Berechnung als Processing Algorithmus implementiert haben
M4: Z Berechnung	14.05.2020	Alle Werte können berechnet werden und sind als Processing Algorithmen implementiert
M5: Gewichtetes Z	28.05.2020	Der gewichtete Wert Z kann berechnet werden
M6:Dokumentation fertig	10.06.2020	Die Dokumentation ist komplett fertig gestellt.

Tabelle 11.2. Definierte Meilensteine

11.4. Rollen

Name	Rolle	Interesse
Prof. Stefan Keller	Betreuer	Ein produktives Endprodukt, das gut für die Weiterentwicklung dokumentiert ist, eine erfolgreiche Bachelorarbeit
Yves Maurer	Industriepartner	Ein produktives Endprodukt
Joël Schwab und Ryan Horiguchi	Entwicklungsteam	Eine gute Bachelorarbeit, ein produktives Endprodukt

Tabelle 11.3. Rollen

11.5. Risikomanagement

ID	Risiko	Schaden	Wahrscheinlichkeit
1	Berechnung wird nicht verstanden	16h	15%
2	Probleme bei der Entwicklung mit QGIS	30h	25%
3	Parallelisierung	4h	10%
4	Input Daten nicht öffentlich zugänglich	Projekt nicht möglich	5%

Tabelle 11.4. Risikotabelle

11.5.1. Umgang mit Risiken

ID	Massnahme
1	Anfragen für Hilfe bei Experten, wie zum Beispiel den Autoren des Algorithmus
2	Kommunizieren mit QGIS Entwicklern oder alternative Wege für die Implementierung finden
3	Alternative Möglichkeiten finden
4	Input Daten vom Bund oder anderen Organisationen bekommen

Tabelle 11.5. Umgang mit Risiken

11.6. Qualitätssicherung

11.6.1. GitLab CI/CD

Für die Qualitätssicherung wird GitLab CI/CD [18] verwendet. Das CI/CD ist ein Teil der Versionskontrolle, mit deren Hilfe man bestimmte Aktionen ausführen kann, wenn bestimmte Bedingungen zutreffen. Sie wird über das File **.gitlab-ci.yml** gesteuert. Nach jeder Änderung am Code, laufen drei statische Code Analyse Tools, welche helfen Fehler und Unschönheiten frühzeitig zu erkennen.

11.6.1.1. mypy

Mypy [19] ist ein statischer Typchecker für Python, welcher die Vorteile von typisierten Sprachen gibt. Um diesen Checker zu verwenden fügt man Typen zu allen Variablen, Parameter und Rückgabewerten von Methoden hinzu, damit das Tool überprüfen kann, ob überall die richtige Werte und Typen übergeben werden und, ob nur Methoden und Variablen aufgerufen werden, die wirklich existieren.

11.6.1.2. flake8

Flake8[20] überprüft den Code für Programmierfehler und stellt sicher, dass der Code PEP 8[21] compliant ist. Bei diesem Checker wurde die maximale Zeilenlänge auf 140 erhöht.

11.6.1.3. Pylint

Pylint [22] prüft den Code nach Programmierfehler, ob Code Standards eingehalten werden, erkennt Code Smells und schlägt dann Möglichkeiten für das Refactoring vor. QGIS ignoriert manche Standard Python Konventionen, wie z. B. die Nennung von Variabeln und Methoden. Aus diesem Grund mussten gewisse Checks ausgeschaltet werden.

11.6.2. Git Feature Branch Workflow

In dem Projekt wurde nach dem Git Feature Branch Workflow [6] gearbeitet. Dies bedeutet, dass für jedes neue Feature ein neuer Feature Branch kreiert wird, welcher erst zurück gemerged wird, wenn die neue Funktionalität fertig implementiert ist. Dies hat den Vorteil, dass der Hauptbranch immer Code enthält, der funktioniert. Um die Qualität zu garantieren, wurde vor jedem merge in den Hauptbranch ein Code Review gemacht.

Projektmonotoring 12

12.1. Soll-Ist-Zeitvergleich

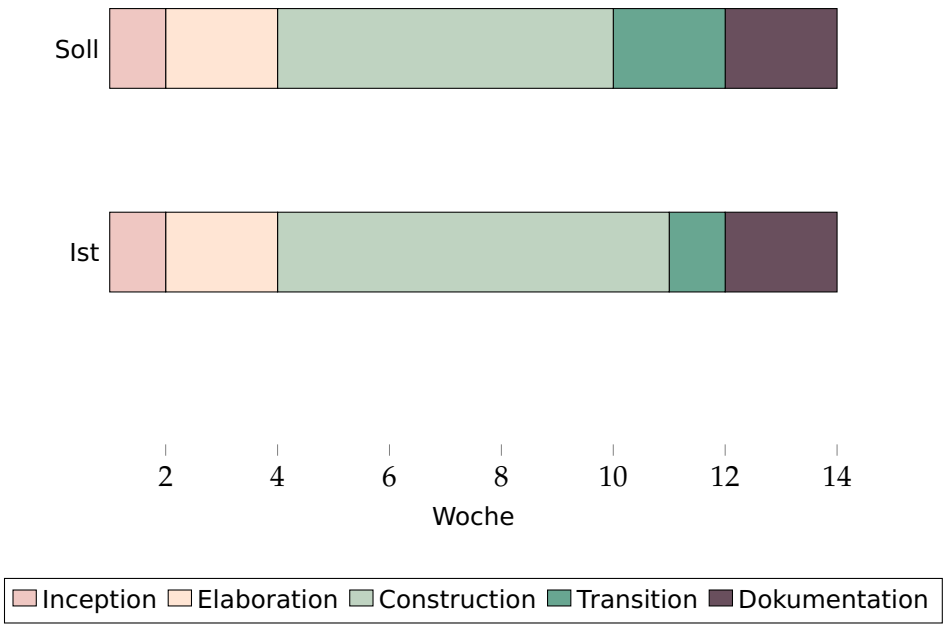


Abbildung 12.1. Zeitverteilung

Die Construction Phase hat sich durch Probleme an der DIS Berechnung und der Parallelisierung verlängert.

12.2. Zeitaufwand pro Person

Pro ETCS Punkt werden 30h Aufwand pro Person erwartet. Der totale Aufwand beträgt somit $12 * 30h = 360h$.

Name	Zeitaufwand
Joël Schwab	368.3h
Ryan Horiguchi	366.8h

Tabelle 12.1. Zeitaufwand pro Person: Stand 10.06.2020

12.3. Codestatistik

Für diese Arbeit haben wurden gesamthaft **1'022** Linien Code geschrieben, ohne Leere Linien und Kommentare sind es **797** Linien.

Es sind **201** GitLab CI/CD Jobs ausgeführt worden, welche durchschnittlich **1 Minute** liefen.

SonarCloud [23] hat dem Code die bestmögliche Qualitätsstufe **A** gegeben. SonarCloud führt eine Vielzahl von statischen Code Analysen durch, welche mögliche Bugs, Sicherheitsschwachstellen und Code Smell erkennt.

Softwaredokumentation 13

13.1. Installation

Um das Tool in QGIS einbinden zu können, muss der Benutzer das Zip File von Gitlab herunterladen.

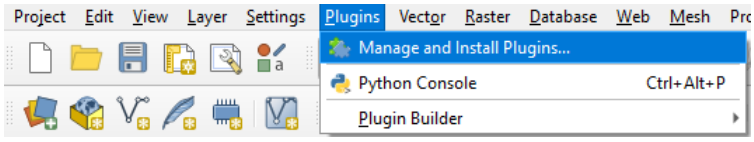


Abbildung 13.1. Plugin Manager öffnen in QGIS [24]

Als nächstes öffnet man QGIS und wählt in der Toolbar den Punkt 'Plugins' aus. In dem Dropdownmenü klickt man auf 'Manage and Install Plugins...'

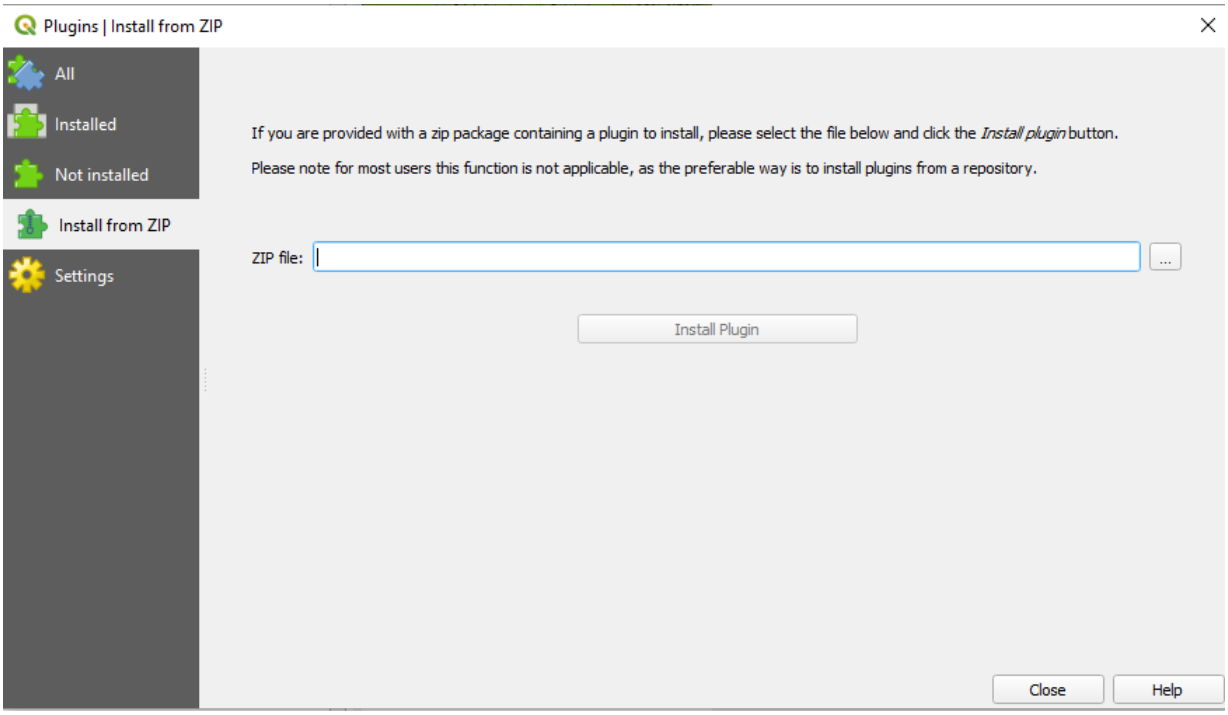


Abbildung 13.2. Pluginmanager in QGIS [24]

Im Plugin Manger wählt man im Menü auf der Seite den Punkt aus ' Install from zip'. Als Zip File wählt man dann ganz einfach das vorher heruntergeladene File aus.

Wenn das Plugin erfolgreich installiert wurde, erscheinen in der Processing Toolbox alle Processing Algorithmen.

13.2. Bedienungsanleitung

13.2.1. Urban Sprawl Clip Raster

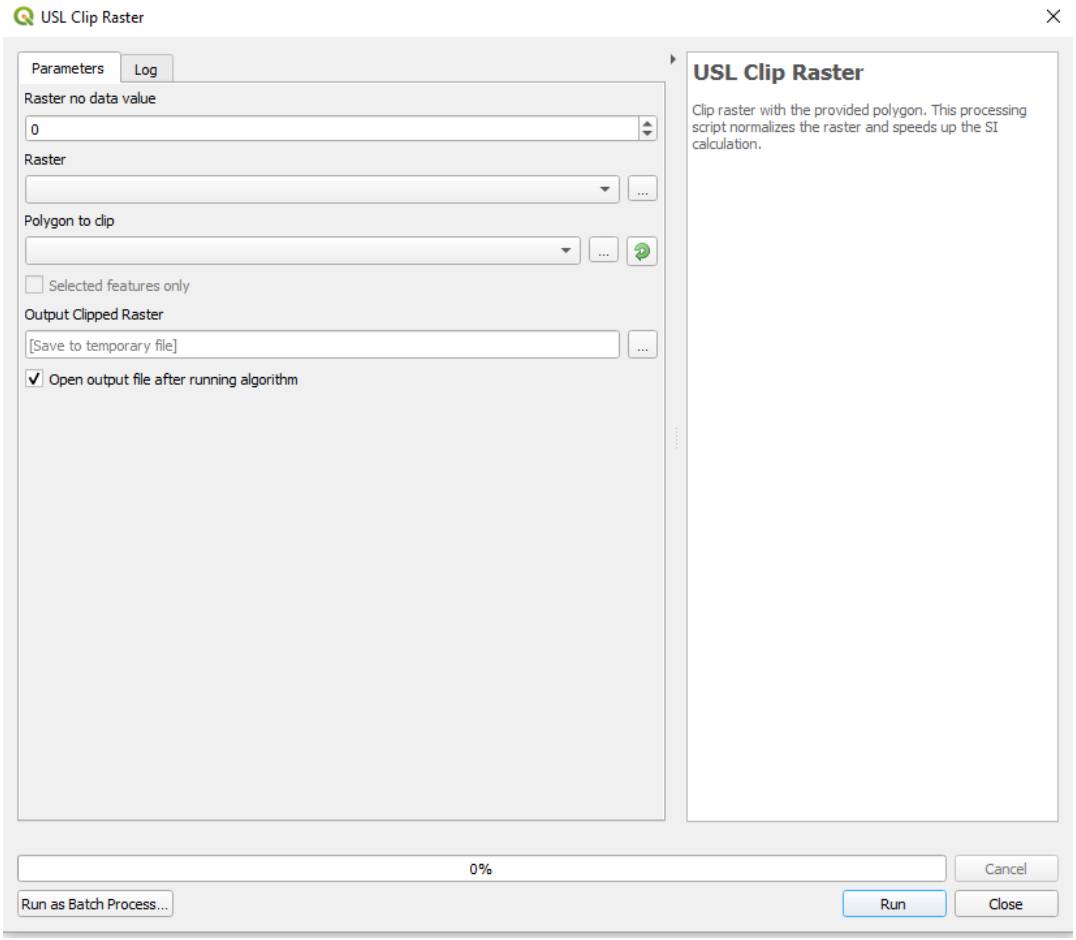


ABBILDUNG 13.3. Urban Sprawl Clip Raster Dialog

Raster no data value: Die Zahl, die beim Raster aussagt, dass die Pixel keine Siedlungs- oder Gebietsfläche sind. Der Default Wert beträgt 0.

Raster: Das Raster, das die Siedlungsfläche beinhaltet. Für eine genauere Berechnung sollte diese Siedlungsfläche über die Gebietsgrenze hinaus gehen.

Polygon to clip: Die Gebietsgrenze, welches analysiert werden sollte. Dabei handelt es sich um ein Vektorpolygon.

Output Clipped Raster: Das ausgeschnittene und überarbeitete Raster, das nur noch die Siedlungsflächen im Gebiet beinhaltet. Allen Werten ausserhalb des Gebiets wird der 'Raster no data value' Wert zugeteilt.

13.2.2. Urban Sprawl SI Calculator

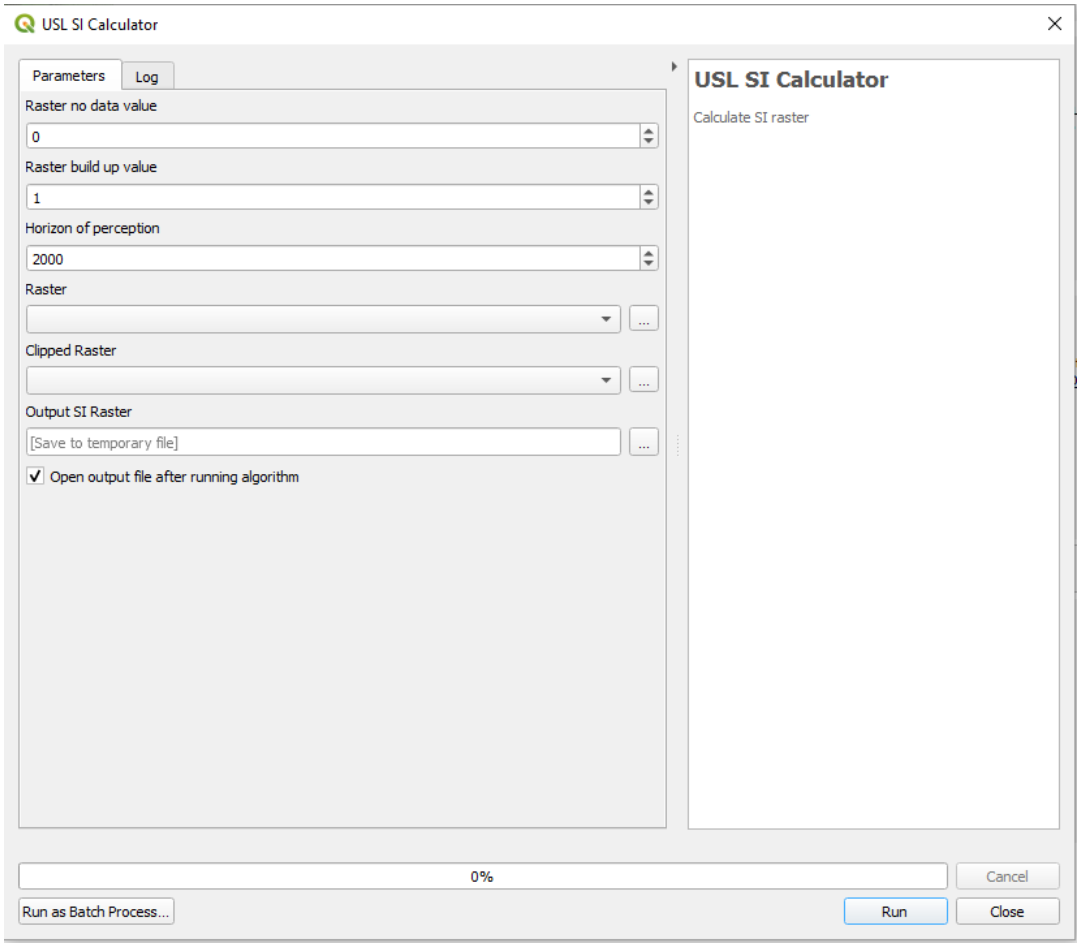


ABBILDUNG 13.4. Urban Sprawl SI Calculator Dialog

Raster no data value: Die Zahl, die beim Raster aussagt, dass die Pixel keine Siedlungs- oder Gebietsfläche sind. Der Default Wert beträgt 0.

Raster build up value: Die Zahl, welche Pixel als Siedlungsfläche markiert im Raster. Der Default Wert ist 1.

Horizon of Perception: Radius, in dem die Siedlungsfläche berücksichtig werden soll. Wichtig zu beachten hierbei ist, dass die Gewichtungsfunktionen auf den Default Wert von 2'000 Meter angepasst sind. Wenn der Radius geändert wird, dann müssen die Gewichtungsfunktionen geändert werden.

Raster: Das Raster, das die Siedlungsfläche beinhaltet. Für eine genauere Berechnung sollte diese Siedlungsfläche über die Gebietsgrenze hinaus gehen. Dies erlaubt für eine genauere Berechnung des SI Werts.

Clipped Raster: Das ausgeschnittene Raster, das vom Urban Sprawl Clip Raster erzeugt wird.

Output SI Raster: Die Dispersion, die für jeden Pixel berechnet wurde. Den Layer kann man einfärben, um sich das ganze grafisch darzustellen.

13.2.3. Urban Sprawl DIS Calculator

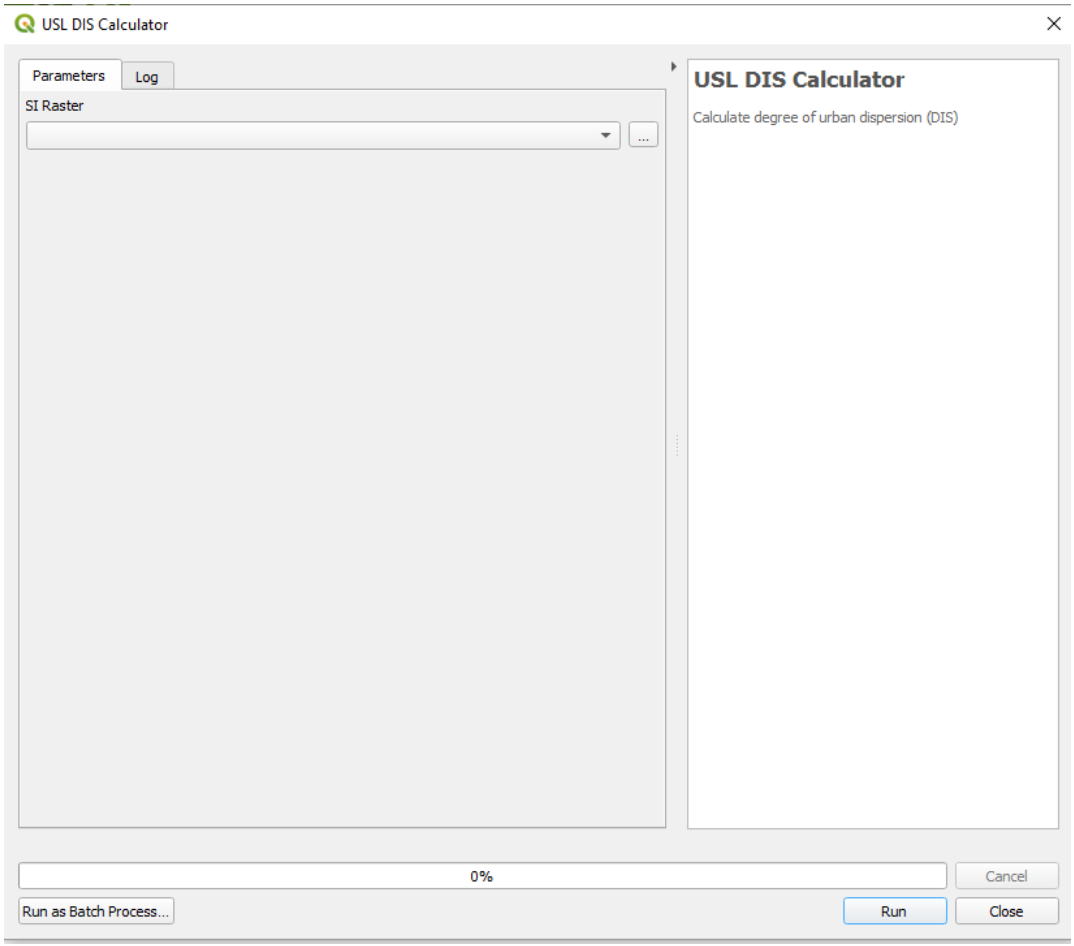


Abbildung 13.5. Urban Sprawl DIS Calculator Dialog

SI Raster: Das Raster, das von Urban Sprawl SI Calculator erzeugt wird.

DIS Wert: Der Dispersions Wert.

13.2.4. Urban Sprawl LUP Calculator

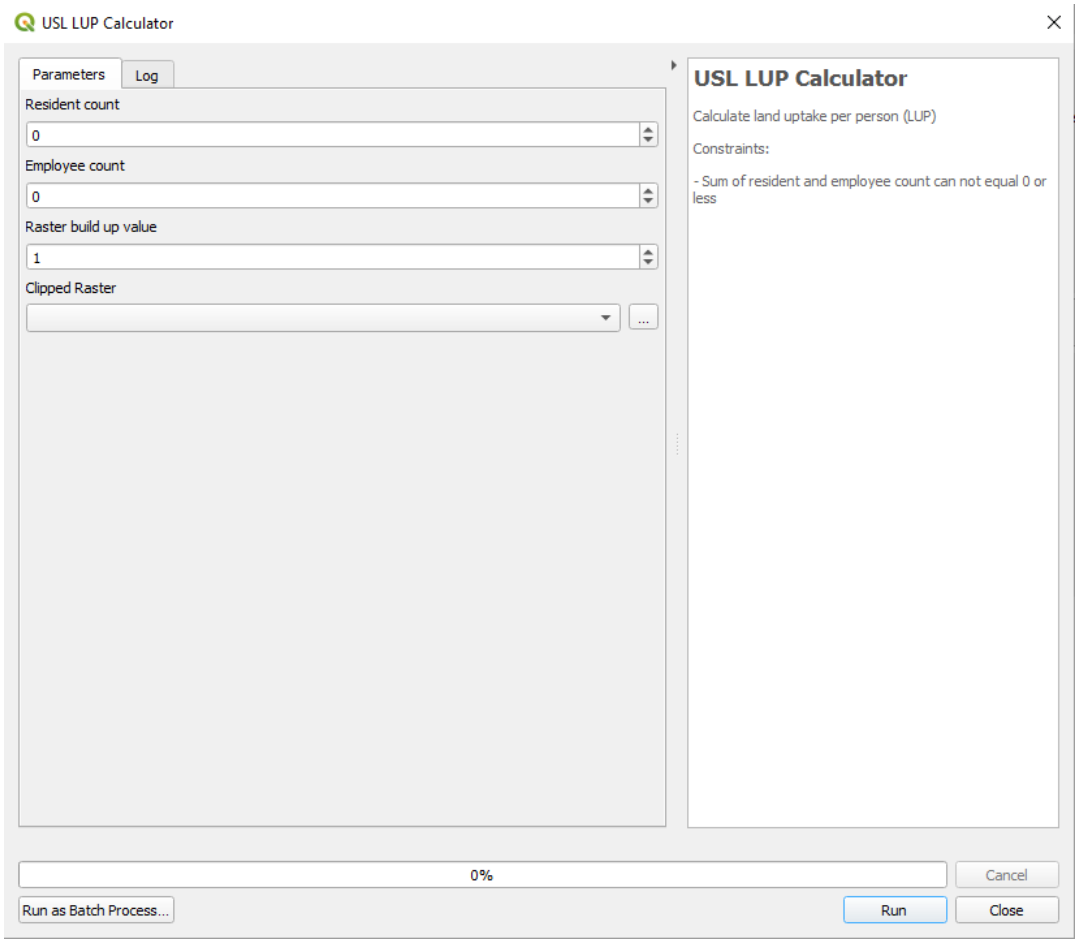


ABBILDUNG 13.6. Urban Sprawl LUP Calculator Dialog

Resident count: Anzahl Einwohner in der Gebietsfläche.

Employee count: Anzahl Arbeitsplätze in der Gebietsfläche.

Es ist auch möglich, die Anzahl der Einwohner und Arbeitsplätze in einen der beiden Inputfelder zu schreiben. Dazu muss das andere Feld 0 enthalten.

Raster build up value: Die Zahl, welche Pixel als Siedlungsfläche markiert im Raster. Der Default Wert ist 1.

Clipped Raster: Das ausgeschnittene Raster, das vom Urban Sprawl Clip Raster erzeugt wird.

FI Wert: Der Wert der Flächeninanspruchnahme.

13.2.5. Urban Sprawl WUP Calculator

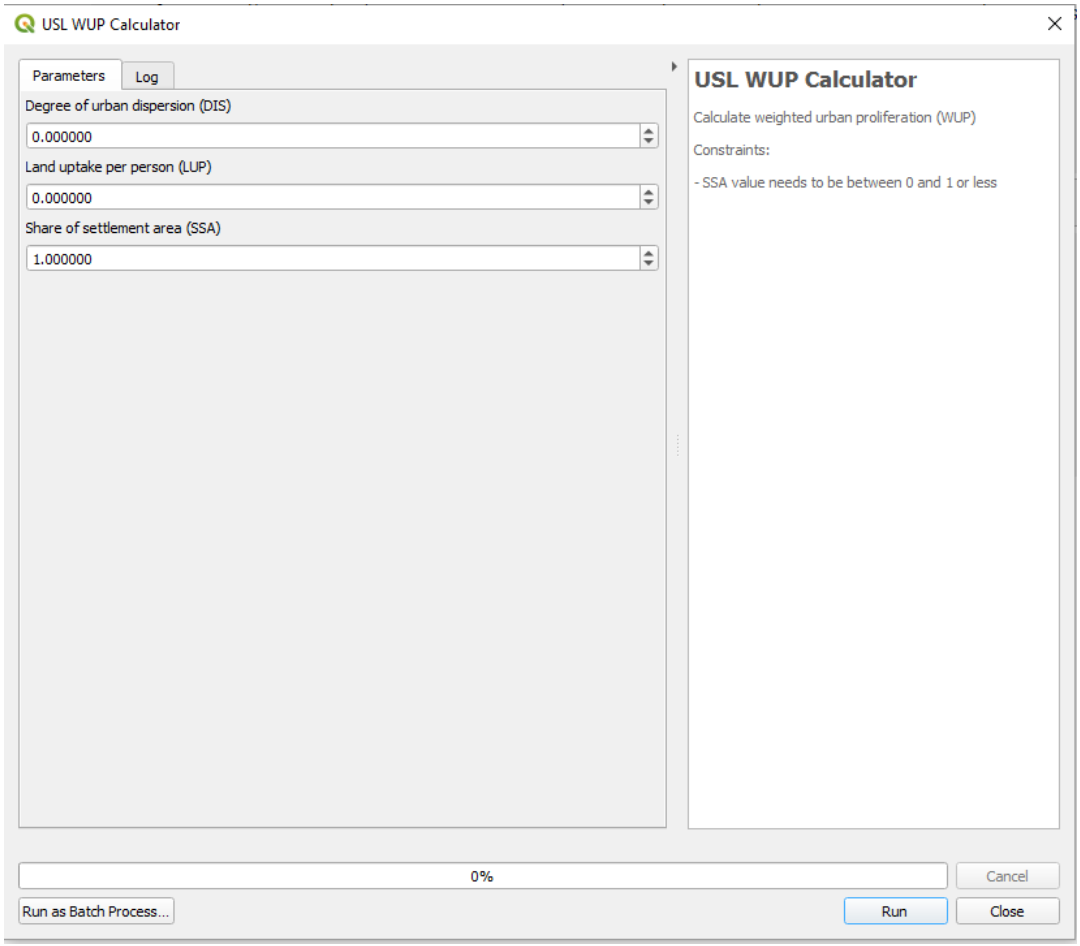


Abbildung 13.7. Urban Sprawl WUP Calculator Dialog

Degree of urban dispersion: Wert der Dispersion.

Land uptake per person: Wert der Flächeninanspruchnahme.

Share of settlement area: Wert, der aussagt, wie gross der Anteil der Siedlungsfläche ist.

Z Wert: Der Wert der Zersiedelung.

13.2.6. Urban Sprawl Urban Sprawl Calculator

Dieser Processing Algorithmus führt die ganze Berechnung in einem Stück durch. Der Nachteil ist es, dass man hier weniger Werte konfigurieren kann. Das SI Raster, das berechnet wird kann man jedoch auch zwischen Speichern und kartografisch darstellen.

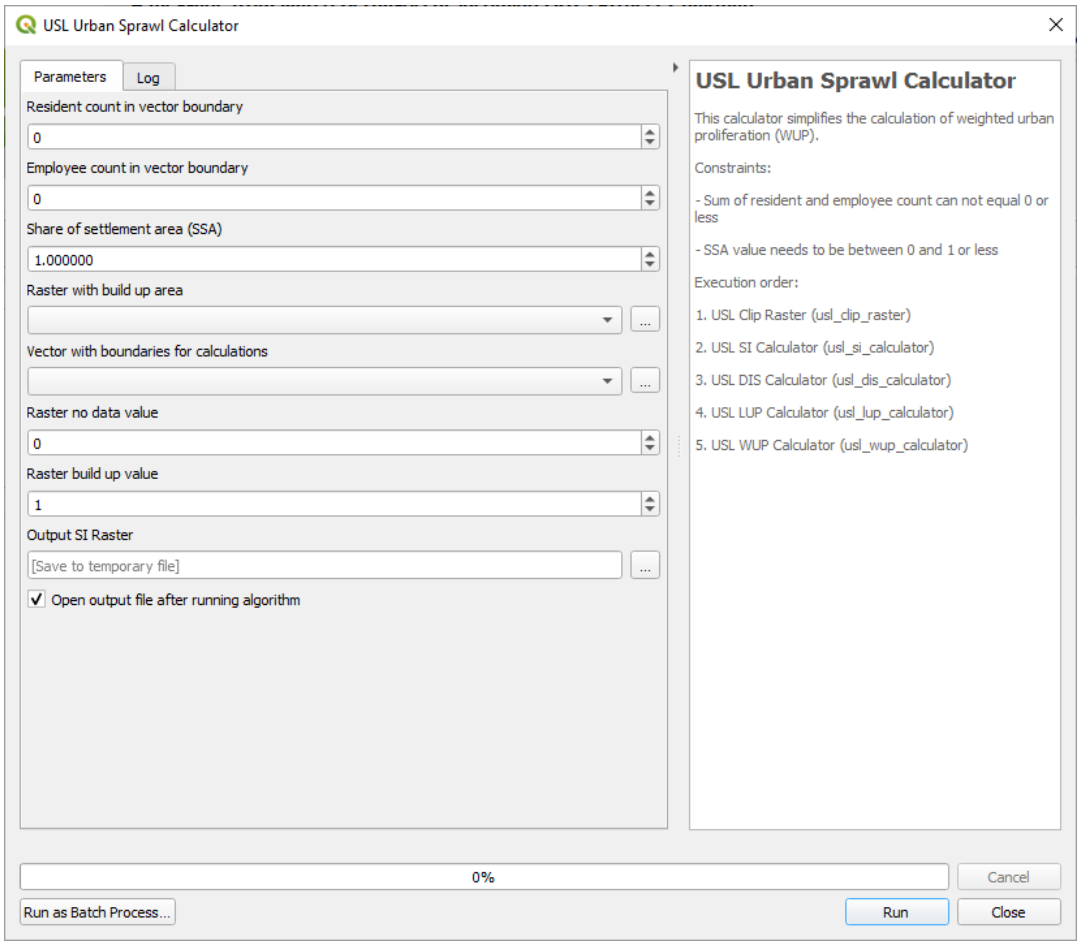


ABBILDUNG 13.8. Urban Sprawl Urban Sprawl Calculator Dialog

Resident count in vector boundary: Anzahl Einwohner in der Gebietsfläche.

Employee count vector boundary: Anzahl Arbeitsplätze in der Gebietsfläche.

Es ist auch möglich, die Anzahl der Einwohner und Arbeitsplätze in einen der beiden Inputfelder zu schreiben. Dazu muss das andere Feld 0 enthalten.

Share of settlement area: Wert, der aussagt, wie gross der Anteil der Siedlungsfläche ist.

Raster with build up area: Das Raster mit der Siedlungsfläche. Die Siedlungsfläche sollte grösser sein als das Vektorpolygon. Dies führt zu einer genaueren Berechnung der Dispersion und daher der Zersiedelung.

Vector with boundaries for calculations: Polygon der Gebietsfläche.

Raster no data value: Die Zahl, die beim Raster aussagt, dass die Pixel keine Siedlungs- oder Gebietsfläche sind. Der Default Wert beträgt 0.

Raster build up value: Die Zahle, welche Pixel als Siedlungsfläche markiert im Raster. Der Default Wert ist 1.

Output SI Raster: Die Pixel in dem Raster, welche die Werte der SI Kalkulation beinhalten. Dieses Raster kann zwischengespeichert werden.

Zersiedelung Z: Als Output kriegt man den Wert für die Zersiedelung Z.

13.3. Weiterentwicklung mit PyCharm

Damit man QGIS exklusive Module nutzen kann, muss man sein PyCharm und PATH Variablen richtig konfigurieren.

13.3.1. Konfiguration unter Windows

Unter Windows kann man sich ein batch/cmd File erstellen, dass die PATH Variablen richtig gesetzt werden [25]. Hier ein Beispiel, wenn man QGIS mit dem OSGEO4W [26] installiert.

```
@echo off

SET "OSGEO4W_ROOT=C:\OSGeo4W64"
SET "PY_CHARM_VERSION=2019.3.3"

call "%OSGEO4W_ROOT%"\bin\o4w_env.bat
call "%OSGEO4W_ROOT%"\bin\qt5_env.bat
call "%OSGEO4W_ROOT%"\bin\py3_env.bat
call "%OSGEO4W_ROOT%"\apps\grass\grass78\etc\env.bat

path "%PATH%;%OSGEO4W_ROOT%\apps\qgis\bin"
path "%PATH%;%OSGEO4W_ROOT%\apps\grass\grass78\lib"
path "%PATH%;%OSGEO4W_ROOT%\apps\Qt5\bin"
path "%PATH%;%OSGEO4W_ROOT%\apps\Python37\Scripts"

set "PYTHONPATH=%PYTHONPATH%;%OSGEO4W_ROOT%\apps\qgis\python"
set "PYTHONHOME=%OSGEO4W_ROOT%\apps\Python37"
set "QT_PLUGIN_PATH=%OSGEO4W_ROOT%\apps\qgis\qtplugins;%OSGEO4W_ROOT%\apps\Qt5\plugins"
set "QT_QPA_PLATFORM_PLUGIN_PATH=%OSGEO4W_ROOT%\apps\Qt5\plugins"

start "PyCharm aware of Quantum GIS" /B "C:\Program Files\JetBrains\PyCharm %PY_CHARM_VERSION%\bin\pycharm64.exe"
```

Teil III.

Anhang

Persönliche Berichte 14

14.1. Joël Schwab

Die Durchführung dieses Projektes war für mich ein sehr spannendes und lehrreiches Erlebnis. Am Anfang des Projektes war es nicht ganz klar, wie der Algorithmus auseht und wie weit wir in der Implementierung der Berechnung kommen. Diese Ungewissenheit war eine neue und grosse Herausforderung, welche die Arbeit sehr interessant gemacht hat. Zuerst mussten wir die Definition des Algorithmus organisieren. Als wir diesen hatten mussten wir ihn genau analysieren. Als wir die Grundlagen des Algorithmus verstanden haben, mussten wir uns entscheiden, welche Teile am wichtigsten waren bevor wir mit der Implementation angefangen haben.

Da kam die nächste Herausforderung. Diese war die Entwicklung einer QGIS Erweiterung. Dies hat für mich die grössten Probleme im Projekt dargestellt. Die Dokumentation war für die meisten Probleme nicht eine grosse Hilfe. Jedoch hat es mir Spass gemacht den Source Code von QGIS genauer anzuschauen, um die richtigen Methoden und Klassen zu finden. Für mich war diese Vorgehensweise in der Entwicklung komplett neu und spannend.

Ich bin zufrieden mit dem Set von Processing Algorithmen, die wir im Rahmen dieses Projekts entwickelt haben. Trotz mehrerer Probleme bei der Entwicklung und auch der Unsicherheit am Anfang haben wir es geschafft die ganze Berechnung zu implementieren.

Die Zusammenarbeit mit Ryan hat sehr gut geklappt. Wir haben schon mehrere Projekte zusammen abgeschlossen, darum wussten wir jeweils, wie die andere Person arbeitet. Die Aufteilung der Aufgaben verlieft also Einwandfrei. Da wir auch ausserhalb der Schule befreundet sind, hatten wir mit der Kommunikation keine Probleme.

14.2. Ryan Horiguchi

Dies war ein spannendes Projekt, das mir den Einblick in ein Open Source Projekt gab, welches verschiedene Herausforderung mit sich brachte.

Diese Arbeit war zu Beginn sehr unklar, da wir beide keine Erfahrung mit QGIS Processing Algorithmen hatten. Zugleich hatten wir nur den C++ Code für die DIS Berechnung. Später haben wird dann noch Literatur zu diesem Thema bekommen, wo der Algorithmus klar beschrieben wurde.

Das Programmieren eines QGIS Processing Algorithmus war eine neue Erfahrung, da man in vielen Fällen nicht genügend Dokumentation fand und daher den Source Code anschauen musste. Man musste auch oft verschiedene Sache probieren, bis man eine Lösung fand, die funktionierte.

Ich bin zufrieden mit dem Resultat. Die 6 Processing Algorithmen, die dem Benutzer ermöglichen die Zersiedelung zu Berechnen. Auch wenn die Implementierung ohne Parallelisierung zustande gekommen ist.

Die Zusammenarbeit mit Joël hat sehr gut geklappt. Da wir beide schon ein paar Projekte

zusammen gemacht haben, kennen wir unsere stärken und schwächen und konnten uns gut ergänzen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur 14

[1] (). „Bundesamt für Raumplanung“, Bundesamt für Raumplanung, Adresse: https://www.are.admin.ch/are/de/home.html (besucht am 06. 04. 2020).

[2] C. Schwick, J. Jaeger, A. Hersperger, R. Muggli und G. Cathomas, *Zersiedelung messen und begrenzen*, 1. Aufl. Haupt Verlag AG, 2018, ISBN: 978-3-258-08086-4.

[3] (). „USM-Toolset“, Swiss Federal Institute for Forest, Snow und Landscape Research WSL, Adresse: https://www.wsl.ch/en/services-and-products/software-websites-and-apps/urban-sprawl-metrics-usm-toolset.html (besucht am 06. 06. 2020).

[4] (). „Swiss Federal Institute for Forest, Snow and Landscape Research“, Swiss Federal Institute for Forest, Snow und Landscape Research, Adresse: https://www.wsl.ch/en/index.html (besucht am 06. 04. 2020).

[5] D. rer. nat. Hans-Georg Schwarz-von Raumer, *CalcDISCellValues*, unpublished, 2018.

[6] (). „Git Feature Branch Workflow“, Atlassian, Adresse: https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow (besucht am 06. 04. 2020).

[7] (). „ArcGIS“, Esri, Adresse: https://www.wsl.ch/en/index.html (besucht am 06. 04. 2020).

[8] O. Zimmermann, „Making Architectural Knowledge Sustainable–Industrial Practice Report and Outlook“, *Carnegie Mellon University*, 2012.

[9] (). „Geospatial Data Abstraction Library“, Open Source Geospatial Foundation, Adresse: https://gdal.org/ (besucht am 06. 04. 2020).

[10] (). „NumPy“, NumPy, Adresse: https://numpy.org/ (besucht am 06. 04. 2020).

[11] C. Schwick, J. Jaeger, R. Bertiller, D. Cavens und F. Kienast, „Urban permeation of landscapes and sprawl per capita: New measures of urban sprawl“, *Elsevier*, 2009.

[12] (). „Euklidischer Abstand“, Wikipedia, Adresse: https://de.wikipedia.org/wiki/Euklidischer_Abstand (besucht am 06. 06. 2020).

[13] (). „QT“, The QT Company, Adresse: https://en.wikibooks.org/wiki/LaTeX/Generating_Bibliographies_with_biblatex_and_biber (besucht am 06. 05. 2020).

[14] (). „QGIS 3.10 Dokumentation“, QGIS, Adresse: https://docs.qgis.org/3.10/en/docs/index.html (besucht am 06. 05. 2020).

[15] (). „QgsTask“, Opengis, Adresse: https://www.opengis.ch/2018/06/22/threads-in-pyqgis3/ (besucht am 06. 06. 2020).

[16] (). „Python 3.7 Dokumentation“, Python Software Foundation, Adresse: https://docs.python.org/3.7/ (besucht am 06. 04. 2020).

[17] (25. Dez. 2018). „python multiprocessing on windows, if __name__ == '__main__'“, Adresse: https://stackoverflow.com/questions/20222534/python-multiprocessing-on-windows-if-name-main (besucht am 06. 05. 2020).

[18] (). „GitLab CI/CD“, GitLab, Adresse: https://docs.gitlab.com/ee/ci/ (besucht am 06. 04. 2020).

[19] (). „mypy“, Adresse: https://github.com/python/mypy (besucht am 06. 04. 2020).

[20] (). „flake8“, Adresse: https://gitlab.com/pycqa/flake8 (besucht am 06. 04. 2020).

[21] (5. Juli 2001). „PEP 8“, Python Software Foundation, Adresse: https://www.python.org/dev/peps/pep-0008/ (besucht am 06. 04. 2020).

[22] (). „Pylint“, Adresse: https://github.com/PyCQA/pylint (besucht am 06. 04. 2020).

[23] (). „SonarCloud“, SonarCloud, Adresse: https://sonarcloud.io (besucht am 06. 06. 2020).

[24] (). „A Free and Open Source Geographic Information System“, QGIS, Adresse: https://www.qgis.org/en/site/ (besucht am 06. 04. 2020).

[25] (). „Silver Spring Energy Consulting LTD.“, Silver Spring Energy Consulting LTD., Adresse: https://silverspringenergy.com/using-pycharm-as-an-ide-for-qgis-3-plugin-development-2/ (besucht am 06. 04. 2020).

[26] (). „OSGeo4W“, OSGeo4W, Adresse: https://trac.osgeo.org/osgeo4w/ (besucht am 06. 04. 2020).

Glossar

ASF Anteil der Siedlungsfläche. 15, 25, 28

DIS Streuung (Dispersion) der Siedlungsfläche. II, III, 15, 27, 28, 32, 42

FI Flächeninanspruchnahme pro Person(Einwohner oder Arbeitsplatz). II, 15, 17, 27, 28, 32, 43

LUP Land uptake per peson (inhabintants and jobs) - Deutsch: FI. III, 43

Pixel Ein Pixel ist ein Punkt in einem Raster. Dieses Quadrat hat eine bestimmt Länge und Breite. Meistens ist diese 15 oder 25 Meter. 26–28, 40, 41, 43, 45, 46

QGIS QGIS ist ein Desktop Geoinformationssystem, das in C++, Qt und Python geschrieben wurde. Es läuft auf allen Betriebssystemen und ist Open Source. II, V, VIII, IX, 3, 7, 9, 13, 14, 17, 23, 24, 28, 29, 31, 35, 39, 46, 49

SI Sprawn at index. II, 9, 15, 19, 26–28, 32, 41, 42, 45, 46, 51

Urban Sprawl Name des Sets von Processing Algorithmen, die im Rahmen dieses Projektes entwickelt wurden. II, III, V, IX, 9, 13, 14, 19, 40–45, 51

WUP Weighted urban proliferation - Deutsch: Z. III, 44

Z Zersiedelung. 3, 4, 9, 15, 17, 27, 28, 44