mundialis GmbH & Co. KG

Sitz u. Registergericht: Bonn
Amtsgericht Bonn HRA 8528

Steuer-Nr.: 205/5823/1574
Ust.-ID-Nr.: DE300200756

Komplementärin:
mundialis Verwaltungsgesellschaft mbH

vertreten durch:
M. Eichhorn

Dokumentation

vom 17.12.2024

Entwicklung eines GRASS GIS Add-Ons zur kollaborativen Erstellung von Trainingsdaten für neuronale Netze

Entwicklung eines GRASS GIS Add-Ons zur kollaborativen Erstellung von Trainingsdaten, verwendbar, um ein neuronales Netz mit Trainingsdaten zu versorgen, als Erweiterung des existierenden Baumerkennungs-Tools.

1.0.0

erarbeitet für:

Regionalverband Ruhr
Referat Geoinformation und Raumbeobachtung
Team Geodaten, Stadtplanwerk, Luftbilder

Kronprinzenstraße 35
45128 Essen

E-Mail: geodaten@rvr.ruhr

Dieses Projekt wird von der Bezirksregierung Münster aus Mitteln des Ministeriums für Umwelt, Naturschutz und Verkehr des Landes NRW gefördert.

erarbeitet von:

mundialis GmbH & Co. KG
Kölnstraße 99
53111 Bonn

Telefon: +49 228 - 387 580 80
Telefax: +49 228 - 962 899 57

E-Mail: info@mundialis.de
Internet: www.mundialis.de

Ansprechpersonen:
M. Metz metz@mundialis.de
M. Eichhorn eichhorn@mundialis.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Änderungen

Tabelle 1: Änderungshistorie des Dokumentes

Datum	Autor:in	Beschreibung
17.12.2024	mundialis	Version v1.0.0

Diese Dokumentation erläutert die notwendigen Arbeitsschritte zur kollaborativen Erstellung von Trainingsdaten auf Basis von True Orthophotos (TOPs) und eines normalisierten Digitalen Oberflächenmodells (nDOM) für ein neuronales Netz. Dieses Netz stellt eine Erweiterung des bereits existierenden Baumerkennungs-Tools der mundialis (entwickelt im Auftrag des RVR) dar, um die Erkennungsmethodik um eben jenes neuronales Netz zu erweitern.

1 Vorbereitung

1.1 Testumgebung

Die Software zur vorliegenden Dokumentation wurde auf einem Linux System mit 16 GB RAM, 16 CPUs und dem GRASS 8.4 Ubuntu GUI Docker Image getestet.

1.2 Skripte und GRASS-Addons herunterladen

Für den Import wird das Import Add-on m.import.rvr verwendet, welches als Teil des rvr_interface GitHub Repository herunterladen werden kann: https://github.com/mundialis/rvr_interface.

Die notwendigen Skripte und GRASS-Addons für die Trainingsdatej können aus dem folgenden GitHub Repository heruntergeladen werden: https://github.com/mundialis/m.neural_network.

Durch Klick auf den grünen „Code“-Button oben rechts kann das Repository entweder für die Nutzung mit der Git-Software geklont werden:

```
# Im Zielverzeichnis ausführen:
$ git clone https://github.com/mundialis/m.neural_network.git
$ git clone https://github.com/mundialis/rvr_interface.git

# bzw. wenn ein SSH-Key auf GitHub hinterlegt wurde:
$ git clone git@github.com:mundialis/m.neural_network.git
$ git clone git@github.com:mundialis/rvr_interface.git
```

oder als ZIP-Datei heruntergeladen und auf dem Rechner entpackt werden. Das Klonen des Repositorys hat den Vorteil, dass es einfach aktualisiert werden kann, während beim Weg über die ZIP-Datei immer das ganze Repository neu herunterzuladen ist.

1.3 Benötigte Software

Das Vorbereiten der Daten und manuell zu labelnden Kacheln für das Erstellen der Trainingsdaten erfolgt in **GRASS GIS**. Das Labeln der vorbereiteten Kacheln ist in **QGIS** vorzunehmen.

GRASS GIS, oder kurz GRASS (*Geographic Resources Analysis Support System*) ist ein kostenloses Open-Source Geographisches Informationssystem (GIS). Mit seinem großen Funktionsumfang wird es für die Verwaltung und Analyse von Geodaten, die Bildverarbeitung, die Erstellung von Grafiken und Karten, die räumliche Modellierung und Visualisierung verwendet. GRASS wird derzeit in akademischen und kommerziellen Einrichtungen auf der ganzen Welt sowie von vielen Regierungsbehörden und Umweltberatungsunternehmen eingesetzt.

1.3.1 GRASS GIS-Basics

Zur Nutzung von GRASS GIS sind einige grundlegende Kenntnisse über die Software nötig.

GRASS GIS unterscheidet sich von anderen GIS-Anwendungen wie ESRI ArcGIS oder QGIS, in denen die Nutzenden sofort verschiedene Daten aus verschiedenen Datenquellen in verschiedenen Projektionen laden und mit der Arbeit an einem Projekt beginnen können. Beim Starten von GRASS GIS müssen die Nutzenden zunächst das Arbeitsprojekt definieren, in dem die GRASS-Sitzung arbeiten soll.

Beim Start von GRASS GIS sind drei Voraussetzungen erforderlich (über den Startbildschirm oder die Kommandozeile einrichtbar):

- **Datenbankverzeichnis:** Ein Verzeichnis auf der lokalen oder Netzwerkplatte, das alle Daten enthält, auf die GRASS GIS zugreift. Dies ist üblicherweise ein Verzeichnis namens „*grassdata*“, das sich im individuellen Home-Verzeichnis befindet.
- ***Project*** *(früher Location)***:** Spielt die Rolle eines "Projekts". Alle Geodaten, die innerhalb eines *Project* gespeichert sind, müssen das gleiche räumliche Koordinatensystem haben (GRASS GIS unterstützt aus verschiedenen Gründen keine *on-the-fly*-Projektion, kann aber beim Import reprojizieren).
- ***Mapset*****:** Enthält aufgabenbezogene Daten innerhalb eines Projekts. Dies unterstützt die Organisation der Daten in logischen Gruppen oder ermöglicht die parallele Arbeit mehrerer Nutzenden am selben Projekt.

Ein weiterer Unterschied zu anderen GIS-Anwendungen ist das Konzept der ***Computational Region*****.** Diese legt für Berechnungen und Analysen von Rasterdaten den räumlichen Bereich fest, für den die Berechnung durchgeführt wird. Das heißt, mit der *Region* wird die Ausdehnung und Auflösung des Ergebnisrasters bestimmt. Die aktive *Region* kann mit dem GRASS-Befehl *g.region* festgelegt werden.

Zum weiteren Verständnis der GRASS GIS-Basics kann zum Beispiel dieses Intro oder diese GRASS Manual-Seite konsultiert werden.

Weitere hilfreiche Links zum Umgang mit GRASS sind:

- GRASS GIS Homepage
- GRASS GIS 8.3 Reference Manual
- GRASS User Wiki Installation Guide
- GRASS GIS Wiki
- GRASS GIS Tutorial

1.3.2 Einrichten und Starten von GRASS in Docker unter Linux und Windows

Im ersten Schritt muss zunächst GRASS GIS eingerichtet werden. Alle Daten, die in GRASS GIS verwendet werden, liegen in einer Datenbank (*grassdb*) in GRASS-spezifischen Formaten. Diese

Datenbank kann als normaler Ordner neu erstellt werden. Im gewünschten Verzeichnis, in dem die GRASS-Datenbank angelegt werden soll (Achtung: diese kann im Laufe der Analyse sehr groß werden!), muss ein neuer Ordner erstellt werden (üblicherweise als *„grassdata“* im Home-Verzeichnis):

```
$ mkdir grassdata
```

Die Erstellung der Datenbank ist nur einmalig zu Beginn notwendig. Besteht bereits eine *grassdb*, kann auch diese verwendet werden.

Zur Nutzung der Baummodule wird GRASS GIS in Docker verwendet. Hierfür muss Docker auf dem System installiert sein. Genutzt wird das offizielle und öffentlich zur Verfügung stehende Docker_Image GRASS 8.4 Ubuntu. Es enthält alle für die Baummodule notwendigen Abhängigkeiten wie Python3, GDAL und PDAL. Dafür muss zunächst ein lokales Docker Image gebaut werden. Dies geschieht im Wurzelverzeichnis des RVR-Interfaces, in dem auch ein Dockerfile liegt. Dieser Schritt muss nur einmal ausgeführt werden:

```
# Navigieren zum Wurzelverzeichnis des RVR-Interfaces:
$ cd /pfad/zum/interface

# Aufbauen des lokalen Docker Images
# Der Punkt am Ende bedeutet, dass Docker nach dem Dockerfile im aktuellen
# Verzeichnis suchen soll:
$ docker build -t rvr_interface:latest .
```

Statt *latest* kann auch das Datum zum Zeitpunkt der Erstellung (*231004* o. ä.) angegeben werden.

Auf Basis dieses Images kann ein Docker-Container gestartet werden. Die Verzeichnisse, hier die GRASS-Datenbank und die Input-Daten, werden dabei mit -v in den Container gemountet:

```
# Starten des Docker-Containers
$ xhost local:*
$ docker run -it --privileged --rm --ipc host \
         -v /pfad/zu/grassdata:/grassdb \
         -v /pfad/zu/Eingangsdaten:/mnt/data \
         -v "/tmp/.X11-unix:/tmp/.X11-unix:rw" \
         --env DISPLAY=$DISPLAY \
         --device="/dev/dri/card0:/dev/dri/card0" \
         rvr_interface:latest bash

# Oder alternativ
$ xhost local:*
$ docker run -it --privileged --rm --ipc host \
         -v /pfad/zu/grassdata:/grassdb \
         -v /pfad/zu/Eingangsdaten:/mnt/data \
         -v "/tmp/.X11-unix:/tmp/.X11-unix:rw" \
         --env DISPLAY=$DISPLAY \
         --device="/dev/dri/card0:/dev/dri/card0" \
         rvr_interface:231004 bash
```

Für **Windows** muss vorher sichergestellt sein, dass erst Docker Desktop und VcXsrv Windows X Server installiert und eingerichtet sind. *VcXsrv Windows X Server* kann mit den folgenden Schritten installiert und eingerichtet werden:

1. Download und installieren von VcXsrv Windows X Server
2. Starten von Xlaunch und dann konfigurieren (siehe auch hier):
   - Im "Extra Settings"-Fenster Haken setzen bei "Disable access control"
   - Im "Finish Configuration"-Fenster auf "Save configuration" klicken und die Konfiguration z.B. auf dem Schreibtisch/ Desktop speichern

Dann kann mit den folgenden Befehlen der Docker gestartet werden:

```
# eigene IP Adresse herausfinden (verwenden des Wertes bei „IPAddress“ z.B.
10.211.55.10 und nicht 127.0.0.1)
$ Get-NetIPAddress
# oder
$ ipconfig

# Setzen der DISPLAY-Variablen (dabei <YOUR-IP> setzen)
$ set-variable -name DISPLAY -value <YOUR-IP>:0.0

# Starten des Dockers mit den mounts der Daten
$ docker run -it --privileged --rm --ipc host -v C:/Users/path/to/
grassdata:/grassdb -v C:/Users/path/to/rvr_daten:/mnt/data --env
DISPLAY=$DISPLAY --device="/dev/dri/card0:/dev/dri/card0"
rvr_interface:latest bash
```

Es ist nicht notwendig, das Docker Image als root (sudo docker run …) zu starten. Für weitere Rechte können die Nutzenden der Gruppe „docker“ hinzugefügt werden:

```
# Nutzende der Gruppe „docker“ hinzufügen
$ sudo usermod -a -G docker <username>

# überprüfen mit
$ id
```

Solange GRASS GIS innerhalb des Docker-Containers genutzt wird, werden die Pfade im Docker anstelle der lokalen Pfade verwendet. Nach dem obigen Einmounten sind das die folgenden Stammpfade:

- Pfad zur grassdb: */grassdb/…*
- Pfad zu den Input-Daten: */mnt/data/…*

Diese müssen also beim Durchgehen der Schritte aus der Dokumentation ggf. angepasst werden.

Zunächst wird ein neues ***Project*** mit dem EPSG-Code 25832 (Projektion: ETRS89 / UTM zone 32N) angelegt. Alle darin importierten Datensätze verfügen dann über dieselbe Projektion:

```
# Allgemeines Schema zum Erstellen eines neuen GRASS GIS Projects
$ grass -c epsg:epsg-code /grassdb/neuer_project_name

# Erstellen eines neuen GRASS GIS Projects mit dem Namen „project_25832“
$ grass -c epsg:25832 /grassdb/project_25832
```

Im selben Terminal wird nun GRASS GIS gestartet, hier erfolgen die weiteren Schritte. Die GUI öffnet sich normalerweise automatisch. Die weiteren Schritte können entweder im Terminal oder in der GUI erfolgen. Auch in einer geöffneten GRASS-Sitzung können weiterhin alle Terminal-Befehle ausgeführt werden (z. B. Verzeichnisse wechseln, *bash*-Skripte starten, etc.). Falls die GRASS-GUI sich nicht automatisch öffnet oder geschlossen wurde, kann sie mit folgendem Befehl aufgerufen werden:

```
$ g.gui
```

Innerhalb der GRASS-*Projects* gibt es ***Mapsets***, die die Sammlung von thematisch zusammenhängenden Datensätzen ermöglichen. Bei der Erstellung eines neuen *Projects* wird automatisch das PERMANENT-***Mapset*** erstellt. Es ist jedoch sinnvoll, ein selbst benanntes **neues *Mapset*** zu erstellen:

```
# Neues Mapset namens “Gelsenkirchen_2020” erstellen und dort hinein wechseln
$ g.mapset -c Gelsenkirchen_2020

# TIPP: Für alle GRASS-Befehle können Sie <Befehl> --help ausführen, um
# mehr über die Benutzung des Befehls zu erfahren, z.B.:
$ g.mapset --help
```

Alternativ kann auch direkt beim Starten von GRASS GIS ein neues *Mapset* erstellt werden:

```
# Allgemeines Schema zum Erstellen eines neuen Mapsets in einer bestehenden
# Location
$ grass -c /grassdb/location_name/neuer_mapset_name

# Erstellen eines neuen Mapsets „Gelsenkirchen_2020“ in dem Project „locati-
on_25832“
$ grass -c /grassdb/location_25832/Gelsenkirchen_2020
```

Nun befinden sich die Nutzenden in GRASS GIS innerhalb des (neuen) *Projects* (im Beispiel: “location_25832”) und des neu angelegten *Mapsets* (im Beispiel: „Gelsenkirchen_2020“). Von hier aus können die im weiteren Verlauf in GRASS GIS stattfindenden Schritte ausgeführt werden.

Der folgende Befehl öffnet GRASS GIS zu einem anderen Zeitpunkt in einem bestimmten *Project* und *Mapset* erneut:

```
# Allgemeines Schema zum Öffnen von GRASS in bestimmten Mapsets
$ grass /pfad/zu/grassdata/yourlocation/yourmapset

# Wiederaufnahme einer GRASS-Sitzung in dem Project “location_25832” und dem
# Mapset “Gelsenkirchen_2020”
$ grass /grassdb/location_25832/Gelsenkirchen_2020
```

Zum Umbenennen oder Löschen eines bestehenden *Mapsets* kann direkt der Ordner des *Mapsets* in der *grassdb* umbenannt bzw. gelöscht werden, z. B.:

```
# Allgemeines Schema zum Umbenennen/Löschen von GRASS Mapsets
$ mv /pfad/zu/grassdata/yourlocation/yourmapset /pfad/zu/grassdata/yourloca-
tion/new_name
$ rm -rf /pfad/zu/grassdata/yourlocation/yourmapset
```

Alle GRASS GIS entwickelten Addons, die für die Einzelbaumerkennung mittels Neuronalem Netz benötigt werden, haben den Key „neural network“ und können mit diesem in der GRASS GIS GUI gesucht werden.

1.3.3 Ändern der Eigentumsverhältnisse der erstellten GRASS-Daten

Der GRASS GIS Docker-Container läuft mit Root-Rechten. Daher werden die im Docker erstellten Daten als root erstellt, weshalb das Zugreifen auf diese Daten nicht wie gewohnt von allen Nutzenden möglich ist. Um z. B. eine im Docker erstellte GRASS GIS *Project/Mapset* von außerhalb des Dockers öffnen zu können, müssen vorher ggf. die Nutzenden und die Gruppe außerhalb des Dockers angepasst werden. Dies kann z. B. über die folgenden Befehle erfolgen:

```
# Abfrage der Rechte auf die gesamte GRASS-Datenbank
$ ls -lah /pfad/zu/grassdata
drwxr-xr-x 14 root root   4,0K Jan 31 13:43 project_erstellt_im_docker
drwxrwxr-x 5 USER GRUPPE 4,0K Jan 24 15:36 project_erstellt_außerhalb_docker

# Anpassen der Nutzenden und der Gruppe (beide können aus der Antwort zu
# dem obigen ‚ls‘ an dem anderen Project entnommen werden und
# dementsprechend gesetzt werden)
$ sudo chown USER:GRUPPE -R /pfad/zu/grassdata/project_erstellt_im_docker
```

1.4 Benötigte Datensätze

Für das Erstellen der Trainingsdaten werden die folgenden Datensätze benötigt:

- Gebiet:
  - Es wird ein Gebiet im Vektordatenformat (z. B. Shapefile, Geopackage) benötigt, für das die Prozessierung ausgeführt werden soll.
  - Alle anderen Datensätze müssen dieses Gebiet vollständig abdecken.
- TOPs:
  - Die benötigten True Orthophotos (TOPs) im .tif Format mit vier Farbkanälen (RGBI) müssen innerhalb eines gemeinsamen Verzeichnisses vorliegen (in diesem Verzeichnis dürfen keine weiteren Dateien enthalten sein).
- Punktwolken:
  - Die benötigten Punktwolken im .laz Format müssen in einem gemeinsamen Ordner vorliegen (in diesem Verzeichnis dürfen keine weiteren Dateien enthalten sein).
- ggf. Digitales Geländemodell (DGM; engl. digital terrain model (DTM)):
  - Falls nicht das NRW-DGM1 zur nDOM-Erstellung genutzt werden soll, kann auch ein individuell definiertes DGM mit einer idealen Auflösung von 0,5 - 1,0 m verwendet werden. Dieses muss in einem GDAL-konformen Format vorliegen (z. B. GeoTIFF, ASCII XYZ).

1.4.1 Import und Aufbereitung der Datensätze in GRASS GIS

Für den Import und die Aufbereitung der Eingangsdaten wurde das GRASS-Addon *m.import.rvr* entwickelt. Dieses fasst mehrere Arbeitsschritte zum Datenimport und zur Aufbereitung zusammen, wobei auch die Möglichkeit besteht, Gebäudedaten von OpenNRW bei Bedarf automatisch herunterzuladen. Das Import-Tool kann ebenso für das Gebäude-Tool (s. Dokumentation „Gebäude- und Dachbegrünungsdetektion“) genutzt werden. Nachfolgend findet sich eine Übersicht der Aufgaben des Import-Addons:

- Importieren der benötigten Datensätze
  - Gebiet, TOPs, 2,5D-Punktwolken, DGM
- Aufbereitung der TOPs
  - Importieren der einzelnen TOP-.tif-Dateien, die in dem angegebenen Gebiet liegen, und Resampling dieser auf 0,2 m Auflösung
  - Zusammenfassen der einzelnen TOP-Tiles zu einem virtuellen Raster (VRT). Jeder Kanal liegt als eigene Rasterkarte (*top_red_02*, *top_green_02*, *top_blue_02* und *top_nir_02*) im Mapset. Diese VRT-Karten setzen sich wiederum aus den einzelnen TOP-Tiles zusammen, z. B *top_red_02* aus verschiedenen anderen *top_*_02*.
- Aufbereitung der 2,5D-Punktwolken
  - Importieren der 2,5D-Punktwolken in GRASS GIS und Verarbeitung zu einem Digitalen Oberflächenmodell (DOM; engl. *digital surface model* (DSM)) im Rasterformat mit 0,5 m Auflösung.
  - Importieren der .laz-Tiles als DOM-Raster und anschließend Zusammensetzung zu einem Gesamtraster. Dies geschieht mit Hilfe des Import-Addons, das intern das Addon *r.in.pdal.worker* aufruft, um die einzelnen .laz-Dateien auch parallel importieren zu können.
- Berechnung des normalisierten Digitalen Oberflächenmodells (nDOM; engl. *normalised digital surface model* (nDSM))
  - Zur Berechnung des nDOM (der Differenz aus DOM und DGM) wird das Addon *r.import.ndsm_nrw* im Import-Addon aufgerufen. Dieses interpoliert eventuelle NoData-Lücken im importierten DOM, die durch zu geringe Dichte der Punktwolken auftreten können, berechnet aus vorhandenem DOM und einem DGM das nDOM und resampled es auf die aktuelle *Region*.
  - Zur Berechnung des nDOM wird entweder ein individuell definiertes DGM oder das NRW-DGM verwendet. Zur Nutzung eines eigenen DGMs kann der Parameter *dsm_dir* im Import-Addon angegeben werden. Wird kein DGM definiert, bezieht das Addon automatisch die benötigten Tiles des NRW-DGM.
  - Das DGM für das Verbandsgebiet wird sowohl vom RVR als auch vom Land NRW gekachelt im XYZ-Format zur Verfügung gestellt. Dabei ist zu beachten, dass sich die XY-

Koordinaten hier nicht wie allgemein üblich auf die Pixelmitte beziehen, sondern auf die linke untere Ecke eines Pixels. Wenn dies nicht beachtet wird, kann es zu einem Versatz um einen halben Pixel kommen, außerdem passen die Grenzen dann nicht mehr zu den Grenzen der TOP-Kacheln.

- Bei der Nutzung eines individuell definierten DGMs wird dessen Versatz wie folgt gehandhabt:
  - Bei der Nutzung von XYZ-Tiles wird der Versatz vom Import-Addon korrigiert (Auflösung des DGMs muss angegeben werden).
  - Bei der Nutzung eines GeoTiffs wird von einer vorher erfolgten Korrektur ausgegangen und das GeoTiff „*as is*“ eingeladen.

Im Folgenden werden die Parameter des Import-Addons genauer aufgeschlüsselt, die für die Einzelbaumerkennung relevant sind:

Eingangsparameter:

- ***type***: Analysetyp, für den die benötigten Daten importiert werden sollen: „neural network“.
- ***area***: Pfad zur Vektordatei, die das zu berechnende Gebiet enthält.
- ***dsm_dir***: Pfad zum Ordner, in dem die DOM-LAZ liegen.
- ***dsm_tindex*****:** Pfad zu einem / für einen Tileindex für die DOM LAZ-Dateien, die im *dsm_dir* liegen (optional). Der Tileindex muss ein Attribut *location* haben, in dem der absolute Pfad zu den .laz-Daten (innerhalb des Dockers) steht. Dieser wird entweder verwendet, sofern die Datei existiert und er nicht bei jedem Lauf des Addons erstellt werden muss, oder gespeichert, wenn die Datei nicht existiert, damit er für zukünftige Läufe wiederverwendet werden kann. Das erste Erstellen des Indexes *dsm_tindex* kann lange dauern, weshalb es sinnvoll ist, vor allem diesen Index wiederzuverwenden. Ändert sich hingegen die Auflösung der Eingangsdaten (z. B. zwischen zwei Bildflugjahren von 10 cm auf 7,5 cm), muss dieser neu berechnet werden (optional).
- ***dtm_dir***: Pfad zu einem Ordner mit mehreren XYZ-Dateien des DGMs (Import und Shift, um die Angabe der unteren linken Ecke statt des Pixelzentrums zu korrigieren, hierfür wird die dtm_resolution benötigt) (optional, Leerlassen, um automatisch das DGM von Open.NRW runterzuladen)
- ***dtm_file***: Pfad zur DGM-Datei im Format GeoTiff (Import und Resampling ohne Korrektur) oder XYZ (Import und Shift, um die Angabe der unteren linken Ecke statt des Pixelzentrums zu korrigieren, hierfür wird die *dtm_resolution* benötigt) (optional, *Leerlassen, um automatisch das DGM von Open.NRW runterzuladen*).
- ***dtm_resolution***: Original-Auflösung des DGM, damit bei Verwendung einer DGM-XYZ-Datei oder mehreren DGM-XYZ-Dateien in einem Ordner eine Korrektur des Versatzes durchge-

führt werden kann (optional; notwendig, wenn *dtm_dir* oder *dtm_file* als XYZ-Datei gegeben)

- ***dtm_tindex***: Pfad zum / für einen Tileindex für die DTM-XYZ-Dateien, die im *dtm_file* Ordner liegen. Der Tileindex muss ein Attribut *location* haben, in dem der absolute Pfad zu den XYZ-Daten (innerhalb des Dockers) steht. Dieser wird entweder verwendet, sofern die Datei existiert, damit er nicht bei jedem Lauf des Addons erstellt werden muss, oder gespeichert, wenn die Datei nicht existiert, damit er für zukünftige Läufe wiederverwendet werden kann. Ändert sich hingegen die Auflösung der Eingangsdaten (z. B. zwischen zwei Bildflugjahren von 10 cm auf 7,5 cm), muss dieser neu berechnet werden (optional).
- ***top_dir***: Pfad zum Ordner, in dem die TOP-GeoTiffs liegen.
- ***top_tindex***: Pfad zum / für einen Tileindex für die TOPs, die im *top_dir* liegen. Der Tileindex muss ein Attribut *location* haben, in dem der absolute Pfad zu den TOPs (innerhalb des Dockers) steht. Dieser wird entweder verwendet, sofern die Datei existiert und er nicht bei jedem Lauf des Addons erstellt werden muss, oder gespeichert, wenn die Datei nicht existiert, damit er für zukünftige Läufe wiederverwendet werden kann. Ändert sich hingegen die Auflösung der Eingangsdaten (z. B. zwischen zwei Bildflugjahren von 10 cm auf 7,5 cm), muss dieser neu berechnet werden (optional).
- ***type***: Analysetyp, für den die benötigten Daten importiert werden sollen: „trees analysis“.
- ***-c***: Flag zum Prüfen, ob alle für den Analysetyp erforderlichen Input-Daten importiert werden können, ohne dabei den eigentlichen Import zu starten.
- ***-b***: Flag für den Download der Referenzgebäudedaten von OpenNRW, sofern der entsprechenden Parameter *reference_buildings_file* nicht gesetzt ist (Achtung: Es steht immer nur der aktuellste Datensatz zur Verfügung). Werden hier nicht benötigt!

Ein Teil der Berechnung im Import-Modul kann parallel erfolgen. Für die Konfiguration der Parallelisierung können folgende Parameter gesetzt werden:

- ***memory***: Arbeitsspeicher in MB, der dem Addon zur Verfügung stehen soll (optional, Defaultwert ist 300 MB)
- ***nprocs:*** Anzahl der Kerne für die Parallelprozessierung (optional, Defaultwert ist -2 (Anzahl der verfügbaren Kerne minus 1), für serielle Prozessierung auf 1 setzen)

```
# m.import.rvr ausführen:
$ DATAFOLDER=/mnt/data
$ m.import.rvr memory=300 type=“neural network“ \
   area=${DATAFOLDER}/Gebiet/gelsenkirchen.gpkg\
   top_dir=${DATAFOLDER}/2020_Sommer/TOP/ \
   dsm_dir=${DATAFOLDER}/2020_Sommer/Punktwolke_2_5D_RGBI/ \
   dtm_dir=${DATAFOLDER}/2020_Sommer/DGM/dgm1_gelsenkirchen_2020_correcte-
d.tif
```

Hinweis:
Dieser Schritt kann aufgrund der Größe der Eingangsdateien und gemounteten Datenverzeichnisse viele Stunden in Anspruch nehmen.
Zur Vermeidung von Fehlern empfehlen wir für den Arbeitsspeicher den Defaultwert von 300 MB. Für *memory* erfolgt dann keine Angabe. Darüber hinaus ist es sinnvoll, die maximale Anzahl der verfügbaren Kerne minus 1 anzugeben (Default für *nprocs*), um die Rechenzeit gering zu halten.

Das Import-Addon importiert dann die folgenden Daten für die Einzelbaumerkennung:

- ***study_area:*** eine Vektorkarte des Gebietes
- ***top_red_02, top_green_02, top_blue_02, top_nir_02:*** die Rasterkarten der einzelnen Kanäle des TOP-Mosaiks
- ***top_ndvi_02:*** die Rasterkarte des berechneten und skalierten NDVI
- ***dsm_02:*** die Rasterkarte des importierten DOMs
- ***ndsm:*** die Rasterkarte des berechneten nDOMs

2 Analyse

Das Mutliadd-on *m.neural_network* beinhaltet die Add-ons *m.neural_network.preparedata*, mit wiederum den Workern *m.neural_network.worker_nullcells* und *m.neural_network.worker_export*, sowie *m.neural_network.preparetraining* mit dem Worker *m.neural_network.preparetraining.worker*.

Die Installation des Multi-Addons installiert automatisch die oben genannten Trainingsdaten-Module sowie ihre zugehörigen Worker-Module. Die Worker-Module werden indirekt durch die Hauptmodule aufgerufen und dienen der parallelen Prozessierung über ein Gitter. Durch das sogenannte Tiling erfolgt ein Teil der Berechnung über mehrere Kacheln.

2.1 Trainingsdatenerstellung

2.1.1 Datenvorbereitung für die Erstellung der Trainingsdaten

Mit dem Multiadd-on *m.neural_network* wird der Prozess des Erstellens von Trainingsdaten über fünf Schritte ermöglicht. Die importierten TOPs werden zur Weiterverarbeitung aus GRASS GIS exportiert. Weiterhin werden entweder schon vorher vorhandene Baumkarten (aus dem Einzelbaum-Tool der mundialis) oder eine vorsegmentierte Kachel exportiert. Diese Baumkarte oder segmentierte Kachel ist die Grundlage für den Labellayer der Trainingsdaten. Die Bäume sind in Abhängigkeit der TOPs und des nDOM als solche zu labeln und zu ergänzen. Dies passiert in QGIS.

Das Add-on *m.neural_network.preparedata* exportiert die notwendigen Daten

- TOPs mit vier Kanälen (RGBI) als TIF (*image_tile_xx_xx.tif*)

- nDOM als TIF (einmal mit der Höhe in Metern (*ndsm_tile_xx_xx.tif*) und einmal auf 0 bis 30 m abgeschnitten und dann auf 1 bis 255 skaliert (*ndsm_1_255_tile_xx_xx.tif*), damit das Neuronale Netz die Daten als Input verwenden kann)
- vorhandene Bäume oder Segmente als Labellayer.

als vorbereitende Schritte für das manuelle Labeln der Trainingsdaten und speichert sie zur weiteren Verarbeitung. Ein Anteil der Kacheln wird in den Ordner **train** geschrieben. Sie sind für das Training des neuronalen Netzes vorgesehen und müssen gelabelt werden. Die restlichen Kacheln werden zur Verwendung im späteren Prozessschritt des Anwendens des neuronalen Netzes im Ordner **apply** aufbewahrt. Neben diesen Ordner liegt ein Tileindex (tiles.gpkg) mit Informationen über die vorhandenen Kacheln und ihre Ordnerzuordnung (**train**/ **apply**). Die Ordnerstruktur sieht wie folgt aus.

```
% Ordnerstruktur
├── apply
│   ├── tile_00_00
│   │   ├── image_tile_00_00.tif
│   │   ├── ndsm_1_255_tile_00_00.tif
│   │   └── ndsm_tile_00_00.tif
│   ├── tile_00_02
│   │   ├──...
├── tindex.gpkg
├── tindex.qml
└── train
    ├── tile_00_01
    │   ├── image_tile_00_01.tif
    │   ├── label_tile_00_01.gpkg
    │   ├── label_tile_00_01.qml
    │   ├── ndsm_1_255_tile_00_01.tif
    │   └── ndsm_tile_00_01.tif
    ├── tile_00_05
    │   ├── ...
```

Im Folgenden werden die Parameter des Add-ons *m.neural_network.preparedata* aufgeführt.

Eingangsparameter:

- ***image_band***: Die Namen der Bildraster, z.B. für die RGBI Bänder der TOPs

- ***ndsm***: Name des nDOM Rasters. Default:ndsm
- ***reference***: Name Referenzvektorkarte.
- ***tile_size***: Die Größe der zu kreierenden Kacheln, auf denen gelabelt wird, in der Einheit Zellen und der resultierenden Form tile_size x tile_size. Default: 512
- ***tile_overlap***: Die Überschneidung der Kacheln in Einheit Zellen. Default: 128
- ***segmentation_minsize***: Die minimale Anzahl an Zellen in einem Segment. Default: 80
- ***segmentation_threshold***: Unterschiedlichkeitsgrenzwert zwischen 0 und 1 für die Segmente. Ein Grenzwert von 0 bedeutet, dass nur identische Segmente verschmolzen werden, während 1 eine Verschmelzung aller Segmente zur Folge hat. Default: 0.3
- ***train_percentage***: Der Anteil an allen Daten, die zum Trainieren verwendet werden sollen und im Ordner **train** abgelegt werden. Default: 30
- ***output_dir***: Der Pfad, unter dem die Daten abgelegt werden.
- ***nprocs***: Anzahl der Kerne für die Parallelprozessierung. Default: 1 führt zu serieller Prozessierung.

```
# Beispiel für die Anwendung von m.neural_network.preparedata um TOPs, nDOM
# und Labellayer als Geopackage zu exportieren
$ m.neural_network.preparedata
image_bands=top_red_02,top_green_02,top_blue_02,top_nir_02 ndsm=ndsm
tile_size=512 output_dir=data/
```

2.1.2 Trainingsdaten anpassen in QGIS

Zur Anpassung des Baumlayers sind folgende Schritte auszuführen.

1. Einladen von TOP, nDOM und Labellayer in QGIS
2. Bäume in Labellayer mit Klasse 2 labeln
    1. vorhandene Bäume können angepasst oder gelöscht werden
    2. Segmente können als Bäume mit der Klasse 2 versehen oder gelöscht werden
    3. neue Bäume können über neues Polygon hinzugefügt werden
3. alle verbleibenden nicht-Baum Objekte (nicht Klasse 2 gelabelt) müssen aus dem Layer gelöscht werden, oder mit der „nicht-Baum“ Klasse 1 versehen werden. Dazu kann die Attributtabelle entsprechend gefiltert werden.
4. Die Veränderungen müssen gespeichert werden. Unter Umständen kann ein Entfernen des gespeicherten Layers aus der Layerliste in QGIS notwendig sein, z.B. wenn man die Daten danach auf einen Server hochladen möchte.

Hinweis: Die gesamte Kachel muss gelabelt sein. Alle Bäume müssen als Klasse 2 definiert sein, der Rest wird automatisch als „nicht-Baum“ für das Training verwendet.

2.1.3 Rasterisieren der Label in GRASS GIS

Die gelabelten Layer werden mit *m.neural_network.preparetraining* für das Training des neuronalen Netzes rasterisiert und als TIF exportiert. Zudem werden Virtuelle Raster Tabellen (VRTs) mit den TOP RGBI und dem skalierten nDOM für die Trainings- und Anwendungsdaten erstellt. Hierbei ist es wichtig, dass die Daten zum Trainieren des Neuronalen Netzes jetzt schon an der richtigen Stelle liegen, da in den VRTs die absoluten Pfade zu den Eingangsbändern enthalten sind.

Die Eingangsparameter sind nachfolgend beschrieben.

Eingangsparameter:

- ***input_traindir***: Name des Pfades zu den eingehenden Trainingsdaten (**train**) mit Unterordnern zu den TOPs und Labellayern der einzelnen Kacheln.
- ***input_applydir***: Name des Pfades zu den eingehenden **apply** Daten mit Unterordnern zu den TOPs der einzelnen Kacheln.
- ***val_percentage***: Anteil an Trainingsdaten, die zur Validierung des Netzes verwendet werden. Default: 20
- ***nprocs***: Anzahl der Kerne für die Parallelprozessierung. Default: 1 führt zu serieller Prozessierung.

3 Visualisierung und Export

3.1 Visualisierung in GRASS

Die Ergebnisse lassen sich in GRASS GIS visualisieren. Sollte die Visualisierung außerhalb des Dockers stattfinden, muss gegebenenfalls noch der Besitz des GRASS GIS ***Projects*** angepasst werden (s. Kapitel 1.3.3). Dann kann die GRASS-Session gestartet werden.

```
$ grass /pfad/zu/grassdb/project_name/mapset_name
```

Falls die GRASS-GUI sich nicht mit öffnet, kann sie mit

```
$ g.gui
```

gestartet werden. Zur einfacheren Interpretation ist es hilfreich, sich im Hintergrund das TOP-Mosaik als Echtfarbenkomposit anzeigen zu lassen. Zunächst sollte der Kontrast des TOP-Mosaiks zur Visualisierung angepasst werden:

```
$ i.colors.enhance red=top_red_02 green=top_green_02 blue=top_blue_02
```

Durch Klick auf die Schaltfläche „V*erschiedene Rasterkartenlayer hinzufügen*“ - „*RGB-Karte hinzufügen*“ können die Kanäle für die Anzeigefarben <red>, <green> und <blue> gewählt werden:

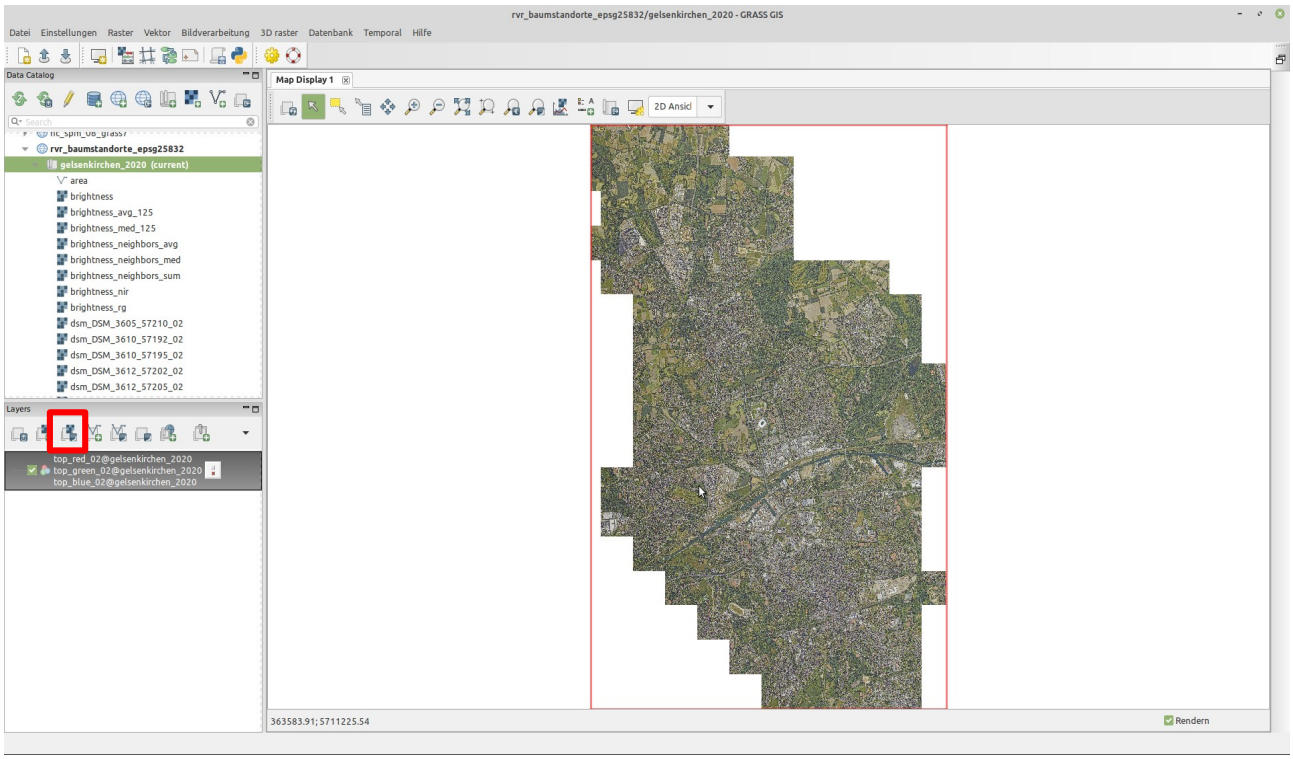

Abbildung 1: RGB-Darstellung des TOP-Mosaiks

3.2 Export der Daten

Für einen ersten Eindruck genügt die Visualisierung in GRASS GIS, die schnelle Anpassung der Darstellung ist jedoch in anderen Geoinformationssystemen wie QGIS oder ArcGIS etwas verständlicher. Zu diesem Zweck sowie zur weiteren Verarbeitung können die Ergebnisse daher zuletzt als gängige Vektordatenformate exportiert werden. Dazu kann das Tool *v.out.ogr* verwendet werden. Dabei muss beachtet werden, dass der Pfad zu den gemounteten Verzeichnissen (z. B. „*/mnt/data*“) angegeben werden muss, wenn im GRASS-Docker gearbeitet wird, damit die Daten nach Schließen des Dockers nicht verloren gehen. Die Dateinamen sollten zur späteren Wiedererkennung und Wiederverwendung den Namen des Gebietes und das Jahr enthalten. Über den ***format***-Parameter kann das gewünschte Dateiformat der ***output***-Datei gewählt werden:

```
# Ergebnis der Einzelbaumdetektion als Shapefile exportieren:
$ v.out.ogr input=tree_objects output=/pfad/zum/output.shp format=ESRI_Shape-
file

# Ergebnis der Veränderungsdetektion als Geopackage exportieren:
$ v.out.ogr input=cd_2020_2022 output=/pfad/zum/output_cd.gpkg format=GPKG
$ v.out.ogr input=tree_objects_2020 output=/pfad/zum/output1.gpkg format=GPKG
$ v.out.ogr input=tree_objects_2022 output=/pfad/zum/output2.gpkg format=GPKG
```

Der Export der Daten kann auch über die GUI erfolgen. Hierfür kann im Suchfeld der Toolbar nach den Modulen „v.out.ogr“ bzw. „r.out.gdal“ gesucht und diese per Doppelklick geöffnet werden. Alternativ können die Module über den Reiter „File“ → „Export vector map“ → „Common export formats [v.out.ogr] “ bzw. „File“ → „Export raster map“ → „Common export formats [r.out.gdal] “ geöffnet werden.

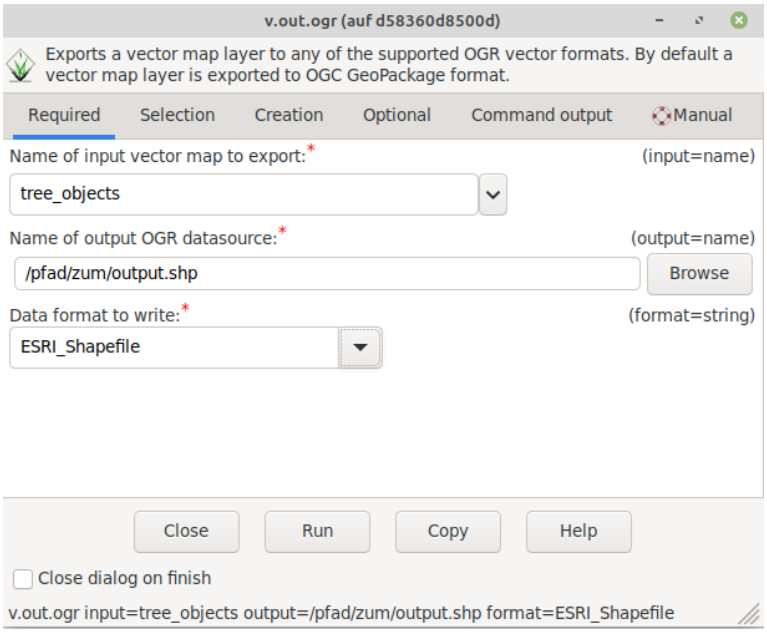

Abbildung 2: Exportieren eines Vektorlayers aus GRASS

In dem Fenster, das sich öffnet, können die verschiedenen Parameter, wie im Kasten beschrieben, gesetzt werden. Pflichtfelder sind mit einem roten Sternchen markiert. Die Bedeutung der Parameter kann der Gesamtdokumentation oder dem Modul eigenen Handbuch entnommen werden (Reiter „Manual“ in Abbildung 2).

4 Umgang mit GRASS Warnungen und Fehlern

Die GRASS-Module geben prinzipiell eher häufig Warnungen aus, um eine eventuelle Fehlersuche zu erleichtern. Diese dienen der Vollständigkeit und können daher normalerweise ignoriert werden. Sollte jedoch ein Modul nicht das gewünschte Ergebnis produzieren, lohnt es sich, die GRASS-Ausgabe auf eventuelle Warnungen hin zu überprüfen. Eine gute Anlaufstelle für mehr Informationen sind außerdem die Handbuch-Seiten der jeweiligen Module. Die Modul-Übersichten, geordnet nach Gruppen (*vector*, *raster*, *general*, etc.), können z. B. über diese Manual-Seite erreicht werden.

4.1 Häufige Fehler und Warnungen

Im Folgenden werden häufige Fehler und Warnungen beim Prozessieren von Daten mit GRASS GIS sowie der Umgang mit ihnen beschrieben. Darüber hinaus lohnt es sich bei Fehlern, insbesondere beim Im- und Export, auch in den Handbuch-Seiten der Module nach weiteren Optionen zu schauen, z. B. das Aktivieren von Flags oder Besonderheiten in den *NOTES* oder *EXAMPLES*.

4.1.1 Warnungen zu Packages

Beim Starten von GRASS GIS kann die folgende oder eine ähnliche Warnung auftreten:

```
$ DeprecationWarning: The distutils package is deprecated and slated for re-
moval in Python 3.12. Use setuptools or check PEP 632 for potential alterna-
tives from distutils.dir_util import copy_tree
```

Warnungen dieser Art können ignoriert werden. Sie beziehen sich auf Systemkomponenten (z. B. Python Packages), die zukünftig Änderungen erhalten werden. Dies hat keinen Einfluss auf die aktuelle Funktionstüchtigkeit von GRASS.

Speziell obige Warnung sollte bei der Nutzung von GRASS 8 (im Docker) nicht mehr auftreten, da die Kompatibilität bereits gewährleistet wurde. Generell laufen aktuelle GRASS-Versionen mit aktuellen Python-Versionen.

4.1.2 Warnungen beim Installieren von GRASS-Addons

Bei der Installation von GRASS-Addons können solche oder ähnliche Warnungen auftreten:

```
$ g.extension m.analyse.trees url=/src/grass-gis-addons/m.analyse.trees -s
...
WARNING: Unable to create '/usr/local/grass83/docs/html/grassdocs.css':
         '/usr/local/grass83/docs/html/grassdocs.css' and
         '/usr/local/grass83/docs/html/grassdocs.css' are the same file. Is
         the GRASS GIS documentation package installed? Installation
         continues, but documentation may not look right.
Compiling...
Installing...
Updating extensions metadata file...
Updating extension modules metadata file...
WARNING: No metadata available for module 'm.analyse.trees': Unable to
         fetch interface description for command '<m.analyse.trees>'.

         Details: <[Errno 2] No such file or directory:
         'm.analyse.trees'>
Installation of <m.analyse.trees> successfully finished
```

Die Warnungen zur GRASS GIS-Dokumentation während der Installation können ignoriert werden. Hier geht es um Metadaten und Dokumentation, die Funktion der Addons ist dadurch nicht eingeschränkt. Ob die Installation eines Addons erfolgreich verlaufen ist, kann an der letzten Zeile abgelesen werden.

4.1.3 Warnungen beim Nutzen der GUI

Beim Nutzen der Graphischen Oberfläche (GUI) treten mitunter folgende oder ähnliche Fehler auf:

```
$ (wxgui.py:6500): Gtk-CRITICAL * *: 13:46:27.691: gtk_box_gadget_distribute:
assertion 'size >= 0' failed in GtkScrollbar
$ (wxgui.py:6500): Gtk-CRITICAL * *: 14:38:27.866: gtk_box_gadget_distribute:
assertion 'size >= 0' failed in GtkCheckButton
$ (wxgui.py:6851): Gtk-CRITICAL * *: 11:49:20.118: gtk_box_gadget_distribute:
assertion 'size >= 0' failed in GtkSpinButton
```

Warnungen dieser Art können ignoriert werden. Die GUI funktioniert dennoch normal.

4.1.4 Warnungen zu inkorrekten Grenzen, *collapsed areas*, Flächenzentroiden, etc.

```
$ WARNUNG: Anzahl inkorrekter Grenzen: 8
WARNUNG: Vect_get_point_in_poly_isl(): collapsed area
WARNUNG: Kann den Flächenzentroiden nicht berechnen.
WARNUNG: Vect_get_point_in_poly_isl(): collapsed area
WARNUNG: Kann keinen Zentroid für die Fläche 101268 berechnen.
WARNUNG: Vect_get_point_in_poly_isl(): collapsed area
WARNUNG: Kann keinen Zentroid für die Fläche 201928 berechnen.
WARNUNG: Flächen mit Größe 0.0 ignoriert.
WARNUNG: Anzahl inkorrekter Grenzen: 7
WARNUNG: Die Datenbankverbindung für den Layer<1> ist nicht definiert
```

Treten Warnungen dieser Art beim Import von Daten auf, sollten sie **nicht** ignoriert werden. Bei der anschließenden Analyse hingegen können die Warnungen ignoriert werden, da die genutzten GRASS-Module, wie. z. B. *v.patch,* in der Regel auch eine topologische Säuberung beinhalten. Das heißt die Warnungen werden der Vollständigkeit halber ausgegeben und eventuelle Probleme dann direkt behoben. Eine Datenbankverbindung ist für GRASS-Vektorlayer nicht zwingend erforderlich, deswegen kann diese Warnung meist ignoriert werden. Für den Fall, dass Vektor-Attribute fehlen, könnte hier jedoch ein Grund dafür liegen.

4.1.5 Fehler bei der Veränderungsdetektion

Wenn die Veränderungsdetektion mit dem Addon *v.trees.cd* mit sehr vielen parallelen Prozessen, z.B. *nprocs=30*, aufgerufen wird, kann es beim Aktualisieren der Attribute zu einem Fehler kommen:

```
$ v.trees.cd inp_t1=trees_objects_2020 inp_t2=trees_objects
output=trees_herne_2020_2022 nprocs=30
       WARNING: Busy SQLITE db, already waiting for 10 seconds...
       [...]
       WARNING: Busy SQLITE db, already waiting for 60 seconds...
       DBMI-SQLite driver error:
       Unable to fetch:
       database schema has changed

       DBMI-SQLite driver error:
       Unable to fetch:
       database schema has changed

       ERROR: Unable to fetch data from table
       [...]
```

In diesem Fall muss die Anzahl paralleler Prozesse reduziert werden, z.B. auf 10, damit der ganze Prozess erfolgreich durchläuft.

4.1.6 Fehler beim Exportieren als Shapefile

Wenn beim Export als Shapefile der folgende Fehler o. ä. auftritt, ist der Name des Attributs zu lang und das Attribut muss umbenannt werden, sodass der Spaltenname maximal 10 Zeichen beinhaltet. Diese Vorgabe kommt von den ESRI Shapefile DBF- Tabellen-Spezifikationen.

```
$ v.out.ogr in=result_trees out=/pfad/zum/output.shp format=ESRI_Shapefile
ERROR 6: Failed to add field named 'durchmesser'
ERROR: Unable to create column <durchmesser>
```

Anschließend kann der Export z.B. mit folgendem Befehl wiederholt werden:

```
# Umbenennen der Spalte zu new_name
$ v.db.renamecolumn map=result_trees column="durchmesser,dm"

# Wiederholen des Exports
$ v.out.ogr input=result_trees output=/pfad/zum/output.shp format=ESRI_Shape-
file --o
```

4.1.7 Warnung beim Export: features without category were skipped

```
$ v.out.ogr in=result_trees out=/pfad/zum/output.gpkg
...
WARNUNG: 14 features without category were skipped. Features without
         category are written only when -c flag is given.
```

Die Warnung kann ignoriert werden. Es ist empfehlenswert, die *-c*-Flag beim Vektor-Export mit *v.out.ogr* nicht zu setzen. Bei Nutzung der *-c*-Flag wird das Ergebnis nicht das Erwartete sein, da in der *Simple Feature Definition* damit Polygone erstellt werden, die es eigentlich gar nicht gibt. Die

C*ategory* als GRASS-Feature ID ist nicht zu verwechseln mit den Attributen von Features (Polygonen). „*Features without category*“ bezieht sich auf Löcher in Polygonen. Diese haben weder eine eigene ID noch Attribute.

4.1.8 Fehler beim Export der Farbtabelle von Rasterdaten (z. B. DOM, nDOM) als GeoTiff

```
$ r.out.gdal input=ndsm_map output=/pfad/zum/output.tif format=GTiff
Using GDAL data type <Float32>
Die Eingabe-Rasterkarte enthält Zellen mit dem NULL-Wert (no-data). Der
Wert -nan wird verwendet, um NoData-Werte in der Eingabekarte zu
kennzeichnen. Sie können NoData-Wert mit dem Parameter nodata bestimmen.
ERROR 6: /pfad/zu/tiff/nDOM.tif, band 1: SetColorTable() only supported for
Byte or UInt16 bands in TIFF format.
```

Beim Export als GTiff ist es für Raster, die nicht als Byte oder UInt16 Datentyp vorliegen, empfehlenswert, die Farbregeln nicht mit zu exportieren. Dafür kann die *-c*-Flag gesetzt werden. Für maximale Kompatibilität mit anderen GIS-Programmen ist auch die *-m*-Flag empfehlenswert. Die Farbregeln können alternativ mit der *-t*-Flag in eine separate Raster-Attributtabelle geschrieben werden. Informationen zu den Flags finden sich auf der Handbuchseite von r.out.gdal.

Der Export kann z.B. mit folgendem Befehl wiederholt werden:

```
$ r.out.gdal -cmt input=ndsm_map output=/pfad/zum/output.tif format=GTiff --o
```

5 Referenzen

Hier befindet sich eine Übersicht der wichtigsten Referenzen in dieser Dokumentation. Kontaktieren Sie uns gerne bei Fragen oder Problemen.

Kontakt mundialis

Ansprechpersonen:

M. Metz	metz@mundialis.de
A. Weinmann	weinmann@mundialis.de
L. Krisztian	krisztian@mundialis.de
M. Eichhorn	eichhorn@mundialis.de

GitHub Repository mit Skripten und GRASS-Addons

https://github.com/mundialis/rvr_interface

GRASS GIS

GRASS Website

- GRASS GIS Homepage
- GRASS GIS Download

GRASS GIS Manual

- GRASS GIS Manual
- GRASS Basics im Manual

GRASS GIS Wiki

- GRASS GIS Wiki
- GRASS User Wiki Installation Guide

Tutorials

- GRASS GIS Workshop
- Analysing environmental data with GRASS GIS