Dokumentation OpenStreetMap GUI Plotter

Inhalt

[1. Getting started 3](#_Toc46479559)

[1.1 Funktionsumfang 3](#_Toc46479560)

[Daten einlesen 3](#_Toc46479561)

[Namen zuweisen & Sortierung auswählen 3](#_Toc46479562)

[Filtern 3](#_Toc46479563)

[Tabellendarstellung der Daten 4](#_Toc46479564)

[Speichern 4](#_Toc46479565)

[Mapping Options 5](#_Toc46479566)

[Code 6](#_Toc46479567)

[Package Übersicht 6](#_Toc46479568)

[setupUi(self, lcd\_display) 6](#_Toc46479569)

[readfile\_bt(self) 6](#_Toc46479570)

[tablefiller(self) 7](#_Toc46479571)

[cleardata\_bt(self) 7](#_Toc46479572)

[sort\_bt(self) 7](#_Toc46479573)

[filteroptions(self,active) 7](#_Toc46479574)

[findjumps(self, jumpborder) 7](#_Toc46479575)

[savefiltered\_bt(self) 7](#_Toc46479576)

[mapping\_bt(self) 7](#_Toc46479577)

[Weiterentwicklung 8](#_Toc46479578)

# Getting started

Kurze Einführung in die Benutzung der GUI. Für detailliertere Beschreibung siehe Funktionsumfang.

## Datei einlesen

In entsprechendes Feld Dateipfad des csv-Files eingeben. Bzw wenn im selben Ordner wie GUI exe dann einfach Dateinamen mit Dateiendung eingeben. Mit dem „Readfile“-Button Eingabe bestätigen.

## Latitude und Longitude auswählen

Nun muss ausgewählt werden in welchen Spalten die GPS Daten stehen.

## Filtern

Um zu filtern die Filterbedingung auswählen, den entsprechenden Hacken setzen und den „Apply Filter“ Button drücken. Ein Ausschnitt der aktuellen Daten wird in der nebenstehenden Tabelle dargestellt.

## Save Filtered Data

Wenn gefilterte Daten gespeichert werden sollen, gewünschten Dateinamen vergeben und auf „Save filtered Data“ Button klicken

## Map mit Daten abspeichern

Gewünschte Darstellung der Daten (Marker, Cluster Marker, Polyline) wählen und einen Dateinamen vergeben. Dann den „Save-Map“ Button drücken.

# Funktionsumfang

Mit Hilfe der GUI können csv Dateien eingelesen werden. Dann können diese nach belieben mit individuellen Filterausdrücken gefiltert und sortiert werden. Die einzelnen Filter werden im laufe dieser Dokumentation genauer beschrieben. Nach dem Filtern können die gefilterten Daten in eine .xlsx oder .csv Datei gespeichert werden. Auf Wunsch wird zudem eine Datei mit identifizierten Sprungpunkten einer gewissen Sprungweite eingeschrieben werden. Zuletzt können in den Mapping Optionen einige Darstellungsoptionen ausgewählt werden und die Karte als .html File gespeichert werden. Diese Datei kann nun in einem geeigneten Browser geöffnet werden.

### Daten einlesen

Um seine Daten einzulesen, muss in das entsprechende Feld ein Dateipfad für das entsprechende .csv File eingegeben werden. Man kann die Datei auch in den gleichen Ordner wie die GUI Datei legen, dann reicht auch der Name des Daten-Files. Danach muss der „Readfilebutton“ geklickt werden. Wichtig wenn die Datei im csv format ist muss .csv angefügt werden.

|  |
| --- |
|  |
| Abbildung 1: Dateien einlesen |

### Namen zuweisen & Sortierung auswählen

Nun müssen wir die Latitude und Longitude Daten richtig zuweisen. Dazu einfach, wenn nicht automatisch richtig zugeordnet wurde, die entsprechenden Spaltenbezeichnungen auswählen. Für die Sortierung wird standartmäßig der timestamp verwendet. Man kann jedoch auch nach jeder anderen beliebigen Spalte sortieren.

|  |
| --- |
|  |
| Abbildung 2: Grundeinstellungen |

### Filtern

Filtern funktioniert mit dem Ablauf: Filtereinstellung -> Filter anschalten -> Apply Filter Button

Will man die Daten zurücksetzen betätigt man den Undo Filter Button.

Die ersten 6 Filter sind für Daten im „float“ Format. Dazu die entsprechende Spalte anwählen, dann gewünschter Operator und eine Bedingung eingeben.

Die beiden „Timestamp“ Filter sind für das Filtern im jeweiligen timestamp format gedacht und funktionieren analog.

Die „First data row“ und „Cut of data row“ kann die Anzahl der Datenpunkte festgelegt werden. Als Eingabe werden nur Integer akzeptiert.

Der „points who jump over“ Filter, filtert Datenpunkte raus die weiter als Angegeben Springen.

Der „repeating lat lon positions“ Filter, filtert Datenpunkte mit den gleichen lat lon Koordinaten aus.

|  |
| --- |
|  |
| Abbildung 3: Filter |

### Tabellendarstellung der Daten

Die aktuell gefilterten Daten werden in der nebenstehenden Tabelle dargestellt. Dabei werden nur die ersten 100 Zeilen dargestellt. Die tatsächliche Zeilenanzahl wird unter der Tabelle dargestellt. Der „Clear Data“ Button löscht alle momentan geladenen Daten.

|  |
| --- |
|  |
| Abbildung 4: Tabelle |

### Speichern

Beim abspeichern kann man den Separator den Filename und das Format auswählen. Der Filename wird per default aus „ID\_StartDate\_EndDate“ zusammengesetzt. Zusätzlich kann bei Bedarf eine Liste mit Sprungpunkten über einer gewissen Sprunggrenze ausgegeben werden. Durch drücken des „Save Filtered Data“ Buttons kann gespeichert werden.

|  |
| --- |
|  |
| Abbildung 5: Speichern |

### Mapping Options

Hier können Einstellungen gemacht werden die die Erstellung der Karte beeinflussen. Einerseits gibt es diverse Möglichkeiten die Daten darzustellen.

|  |  |
| --- | --- |
| Marker | einfacher Marker, Vorsicht ab einer Menge von 3000 Markern wird die Karte instabil |
| Cluster Marker | eine Leistungssparsamere Möglichkeit seine Daten darzustellen, welche die Daten Punkte gruppiert und bei Bedarf besser aufschlüsselt. Diese Option wird empfohlen um viele Datenpunkte darzustellen |
| Polyline | Verbindet die Datenpunkte in der gewählten sortierreihenfolge |
| Tooltip in Markers | Hier kann ausgewählt werden welche Daten in die Tooltips der Marker geschrieben werden sollen um die Datenpunkte zu identifizieren |
| Mark points who jump over | Hier kann man angeben ob und wenn ja ab welcher Sprungweite zwischen Punkten die Marker rot markiert werden sollen |
| Keep every … Marker | Hier gibt man die Abtastrate der Marker an. Heißt, dass zum Beispiel, dass nur jeder 2. Marker auf der Karte dargestellt wird |
| Maptype | Hier kann zwischen einigen Kartendarstellungen auswählen |
| Mapname | Hier gibt man den Filename des Karten-Files an. Hier kann auch ein Dateipfad angegeben werden unter dem die Karte abgespeichert werden soll |

Der „Save Map“ Button speichert die Karte.

|  |
| --- |
|  |
| Abbildung 6: Map Options |

# Code

Hier soll auf die wichtigsten Komponenten des Codes eingegangen werden. Die GUI wird durch das Package Pyqt5 erstellt. Dieses stellt einen App Designer zur Verfügung, durch den die optische Oberfläche erstellt werden kann. Die GUI ist innerhalb des Codes eine Klasse („Ui\_lcd\_display“). Der Aufbau von Pyqt5 code sowie der Umgang mit dem Designer kann in [Tutorials](http://projects.skylogic.ca/blog/how-to-install-pyqt5-and-build-your-first-gui-in-python-3-4/) nachgelesen werden. Die Funktionalitäten kommen über Klassen-Funktionen hinzu. Diese sollen nun nach und nach erläutert werden.

## Package Übersicht

Pyqt5: Liefert das graphische Interface der GUI.

* [Designer](http://projects.skylogic.ca/blog/how-to-install-pyqt5-and-build-your-first-gui-in-python-3-4/) Tutorial
* Widget Übersicht in „pyqt5\_übersicht\_wichtigste\_widgets.pdf“
* [Spezielle Widgets](https://pythonspot.com/pyqt5/)
* [Einführungsvideo](https://www.youtube.com/watch?v=Vde5SH8e1OQ)
* Siehe auch „ui\_to\_py“ für weg von designer zu python

Folium: Erzeugt interaktive Karte mit den entsprechenden Markern & Linien.

* [Quickstart](https://python-visualization.github.io/folium/quickstart.html#Getting-Started)
* [Optimaler zoom](https://stackoverflow.com/questions/58162200/pre-determine-optimal-level-of-zoom-in-folium)
* [verschiedenes](https://nbviewer.jupyter.org/github/ocefpaf/folium_notebooks/tree/master/) Jupiter Notebooks

Numpy: Liefert einige nützlich mathematische Funktionen

Pandas: Hauptwerkzeug für Sortieren, Filtern, manipulieren und abspeichern von Daten

* [Tutorial](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.loc.html#pandas.DataFrame.loc)

cx\_Freeze: Zum erstellen einer Exe. Pyinstaller funktioniert nur sehr umständlich. Problem mit folium

* Siehe „How\_create\_exe.html“
* [Folium Problem](https://github.com/python-visualization/folium/issues/1081)
* [Mögliche Lösung](https://stackoverflow.com/questions/54836440/branca-python-module-is-unable-to-find-2-essential-json-files-when-running-an-ex)

## setupUi(self, lcd\_display)

Diese Funktion wird durch den Designer erstellt und erzeugt die graphische Oberfläche wie vorher im Designer bestimmt. Am Ende dieser Funktion werden den Buttonfunktionen mit den entsprechenden Buttons über die Pyqt5 Funktion „‘self.graphic‘.clicked.connect(‚self.Buttonfunktion‘)“ die entsprechenden graphischen Oberflächen zugewiesen. Sowie die Kontrollvariablen „self.markerexistance“ sowie „self.dataloaded“ initialisiert die später sicherstellen, dass der User gewarnt wird, wenn keine Daten geladen sind.

## readfile\_bt(self)

Diese Funktion liest unter dem angegebenen Pfad die Daten ein, befüllt die comboboxen mit den Spaltennamen und ruft die „self.tablefiller()“ Funktion. Beim befüllen der comboboxen wurde etwas Aufwand betrieben um wenn möglich eine Vorauswahl für vermutlich gewünschte Spalten zu treffen. Beim einlesen der Daten wird eine interne Kopie der Daten angelegt um diese später auf den Anfangswert zurücksetzen zu können. (self.DF)

## tablefiller(self)

Hier werden die Daten die in der Tabelle angezeigt werden, sowie das LCD Display, welches die Anzahl der Linien anzeigt, aktualisiert.

## cleardata\_bt(self)

Wenn der Clear Data Button geklickt wird, werden alle geladenen Daten gelöscht und die befüllten Comboboxen geleert.

## sort\_bt(self)

Beim betätigen des Sortbuttons wird der Name der Spalte, nach der sortiert werden soll, aus der entsprechenden Combobox ausgelesen. Dann werden die Daten mit der pandas internen „self.sort\_values()“ Funktion sortiert und die Tabelle wird erneut aktualisiert.

## filteroptions(self,active)

Je nach dem welcher Button gedrückt wird werden andere Teile dieser Funktion aufgerufen. Dies wird durch den „active“ Parameter bestimmt. Wenn der „Undo Filter“ Button gedrückt wird, wird der untere Teil der Funktion aufgerufen und die Daten werden auf die eingelesenen zurückgesetzt.

Wenn der „Apply Filter“ Button gedrückt wird, wird der Hauptteil dieser Funktion aktiv. Hier wird eine zusätzliche Funktionsinterne Funktion „filtering(…)“ definiert. Diese ‚übersetzt‘ die Eingaben des Users in die passenden slicing Operationen in Pandas. Nun wird geprüft ob einer der ersten 8 Filter angeschaltet wurde und gegebenenfalls mit der Hilfe der filtering Funktion gefiltert. Der repeat Filter basiert auf der pandas Funktion „.drop\_duplicates()“ und der cutoff Filter bassiert auf einfachen slicing Operationen. Nur für den jumppoint Filter wurde eine seperate klassen-Funktion erstellt.

Nach allen Filteroperationen wird der Index des aktuellen Daten Dataframes wieder fortlaufend durchnummeriert. Bei allen Filterprozessen wird gleichzeitig ein string erstellt der abspeichert was wie gefiltert wurde.

## findjumps(self, jumpborder)

Die Funktion die Sprünge in den Datenpunkten finden soll bassiert auf der numpy Funktion „np.diff()“, welche die Differenzen zwischen allen fortlaufenden Einträgen eines Arrays findet. Daraus wird eine insgesamt zurückgelegte Strecke pro Punkt ermittelt und mit der Jumpborder verglichen. Die so ermitelten Jumppoints werden in einem Dataframe zurückgegeben.

## savefiltered\_bt(self)

Hiermit können die gefilterten Daten in einem .csv oder .xslx File abgelegt werden. Zudem wird bei Bedarf ein . xslx File mit allen Jumppoints erstellt. Die entsprechenden Daten zur Erstellung der Files werden in der GUI abgefragt.

## mapping\_bt(self)

Zu guter Letzt kommt die Funktion die dann alles zusammenbringt und die interaktive Karte erstellt. Zu Beginn wird eine leere Karte initialisiert. Nun müssen die entsprechenden Daten aus der GUI ausgelesen werden. Die Jumppoints option ruft die self.findjumps() Funktion um die entsprechenden Sprungpunkte zu ermitteln und fügt diese dann als rote Marker der Karte hinzu. Als nächstes wird überprüft ob Marker erstellt werden sollen und fügt diese gegebenenfalls hinzu. Analog passiert, dass auch für cluster Marker und die Polyline. Zwischenzeitlich wurde der Zoom der Reichweite der Koordinaten angepasst. Zu Letzt werden die Filtersetting noch als separater Marker der Karte hinzugefügt und die Karte dann abgespeichert.

## Weiterentwicklung

Hier sollen einige Ideen gesammelt werden um die GUI weiterzuentwickeln

* [Map direkt in GUI darstellen](https://stackoverflow.com/questions/58590199/how-to-show-folium-map-inside-a-pyqt5-gui)
* [Kleine circle Marker könnten Stabil sein](https://jingwen-z.github.io/how-to-draw-a-variety-of-maps-with-folium-in-python/)
* [Seite mit vielen Interessanten Ideen](https://learndataanalysis.org/category/python-tutorial/python-pyqt5/page/3/)
* [Drag and Drop File into GUI](https://learndataanalysis.org/implement-files-and-urls-to-listbox-widget-drag-and-drop-function-pyqt5-tutorial/)
* [Autocomplete feature](https://learndataanalysis.org/how-to-implement-autocomplete-feature/)
* [Control what user can enter](https://learndataanalysis.org/how-to-implement-data-validation-to-limit-what-a-user-can-enter-pyqt5/)
* Filedialog
* Keyboard shortcuts für die GUI
* Feature Groups von Folium für verschiedene GPS Tracks in der gleichen Karte
* Feature Groups für mehrere Filter Einstellungen
* Autoauswahl der Lat lon Spalten verbessern
* Tabs für die GUI um die Übersichtlichkeit zu verbessern
* Icon für die GUI
* Tabelle ohne For Schleife befüllen?
* Process Bar oder Colorbutton der Ladestatus anzeigt[Filtern, Speichern, Kartenerstellung]