?isySUR

it's easy!

Manual

Published January 13, 2015

Contents

1 Introduction to isySUR 5

1.1 Quick start 5

1.1.1 isySUR in command line version 5

1.1.2 isySUR in window version 5

1.2 What is isySUR? 6

1.3 Content of isySUR 6

2 Using isySUR 9

2.1 Requirements and installation 9

2.1.1 Requirements 9

2.1.2 Installation 10

2.2 Input and output 10

2.2.1 Space usage rules 10

2.2.2 Configuration file 11

2.2.3 Keyhole Markup Language 11

2.3 Command line and GUI version 11

2.3.1 Command line version 11

Parameters 11

2.3.2 GUI version 12

Parameters 12

The menu 12

KML list 13

2.3.3 Using android 13

3 Technical and implementation details 15

3.1 Development 15

3.1.1 Data usage 15

3.1.2 Data structure 16
3.2 Evaluation **16**

4 Added SURs 17

Quick start
isySUR in command line version
isySUR in window version
What is isySUR?
Content of isySUR

1. Introduction to isySUR

Welcome to isySUR (pronounce isy as ['i:zi]), a tool helping you to find the sphere of influence of a space usage rule (SUR). For starting the command line version please see section 1.1.1. For using the window version please read section 1.1.2. You can also run the window version on your android device. This is described in section 2.3.3. For more details about the program see section 1.2.

1.1 Quick start

First decide if you like to use the command line version of isySUR or if you prefer a graphical user interface. Please proceed with section 1.1.1 or section 1.1.2 depending on your choice.

1.1.1 isySUR in command line version

For the command line version type

```
$ python run_isySUR.py cli {input} {output}
```

where `input` is the path to the input file containing the SURs which areas of influence should be computed and `output` is the path for the resulting area(s) of influence. If the path points to a file, only one kml file will be created, containing all calculated areas. If the path points to a directory a kml file for each SUR will be created as well as one containing all areas.

You can leave out `python` if you have installed the program before. For installation instructions see section 2.1. For additional parameters see section 2.3.1.

For information about input and output file types see section 2.2.

1.1.2 isySUR in window version

For the window version just type

```
$ python run_isySUR.py
```

In Menu you find the possibility to load a SUR input file. After calculating you can save the area(s) of influence.

You can leave out `python` if you have installed the program before. For installation instructions see section 2.1. For additional parameters and more features see section 2.3.2.

For information about input and output file types see section 2.2.

Figure 1.1: isySUR in gui version

1.2 What is isySUR?

isySUR is a program that calculates the sphere of influence of a space usage rule by its coordinates and data from OpenStreetMap. It was developed within the informatiCup, a programming competition for students organized by the German Informatics Society (Gesellschaft für Informatik e.V.). Our team is called isy and consists of Adriana-Victoria Dreyer, Jacqueline Hemminghaus, Jan Pöppel and Thorsten Schodde, four students of the master study course 'Intelligent Systems' at Bielefeld University.

Using isySUR is very easy. You give an input file with the coordinates of the SURs and the sphere of influence is computed. Optionally you can add a configuration file that classifies SURs in those that are applicable only indoor, those that are only outdoor or those that are applicable indoor as well as outdoor (both) to get a better result. In the GUI version the calculated areas are shown on a map and you can choose which ones you want to be displayed. They can be saved into a file to reload them whenever you want to have a look at them.

1.3 Content of isySUR

add pydoc file

In the archive you can find the following data (bold printed files will be installed with the install command):

- data (directory) - examples for space usage rules[1]
- **isySUR** (directory) - the source files for the isySUR Python package
  - gui (directory) - source files for gui version
  - tests (directory) - isySUR was developed using test driven development. In this directory you will find the test files.
- testData (directory) - data for tests and test data sets

[1] see section 4

  - dataOnlyForTests (directory) - Some of the test files need data that is stored in this directory
  - Data.txt - a possible input file
- manual.pdf - this manual
- README.txt
- **run_isySUR.py** - the Python script to run isySUR
- run_tests.py - script that runs all or some tests in isySUR/tests/, used for development
- setup.py - the setup script to install isySUR
- surConfig.cfg - a configuration file that can optionally be used for computation[2]

[2] see section 2.3

Requirements and installation
Requirements
Installation
Input and output
Space usage rules
Configuration file
Keyhole Markup Language
Command line and GUI version
Command line version
GUI version
Using android

2. Using isySUR

In this chapter the usage of isySUR is described in detail. First you can find installation instructions in section 2.1, detailed usage instruction for both program versions are following in section 2.3.

2.1 Requirements and installation

isySUR is written in Python so you get a Python script, which is the main program, and a package with the tools. For running the command line version you need not much more than basic Python. If you prefer to use a graphical user interface you will have to install some more packages.

2.1.1 Requirements

1. **basic requirements:**
   - Python 2.7
   - requests (HTTP library)
   - internet connection

   To run a Python script you need Python. Python comes with most Linux distributions. For the use on Windows machines you will have to install Python manually[1]. isySUR uses data from OpenStreetMap[2] to calculate the sphere of influence. Therefore an internet connection is required. The data transfer in Python is realised with the requests[3] library. You can easily install it using pip[4], a Python installation tool

   ```
   $ pip install requests
   ```

   or see requests web page[3].

[1] get Python here: `https://www.python.org`
[2] `http://www.openstreetmap.org`
[3] `http://docs.python-requests.org`
[4] `https://pip.pypa.io`

2. **additional requirements for the GUI version:**
   - kivy
   - concurrent.futures

   For the graphical user interface kivy is used. Kivy[5] is a cross-platform Python Framework for NUI Development. For installation on Ubuntu we recommend the installation with apt-get as describe on the web page. For use on Windows and starting kivy applications from the command line we recommend to install kivy and pygame (needed for kivy) with the unofficial precompiled binaries[6][7] On MacOSX kivy is a kind of launcher. Therefore installation isySUR is not possible[8].
   On all OS futures can be installed via pip

   ```
   $ pip install futures
   ```

2.1.2 Installation

You got an archive of type .tar.gz. Please unpack the archive into a directory you want. Because isySUR is a Python script you do not need to install it. It is also possible to install only the dependencies and run isySUR through typing:

```
$ python run_isySUR.py [parameters]
```

For installing the script and tool package browse the directory you chose and install isySUR with help of Python:

```
$ python setup.py install
```

Now you are able to start isySUR anytime through

```
$ run_isySUR.py [parameters]
```

2.2 Input and output

The input of the space usage rules was specified in the programming task as well as the output in .kml format. For more information about the decision to add a configuration file please see section 3.1.1.

2.2.1 Space usage rules

The SUR file is a plain text file. The first line holds the number of space usage rules in the file, that will be used for computation. Line-serially the SURs follow. Each of this lines starts with the id of the SUR. Then latitude and longitude follow and the last entry is the name or description of the SUR. The entries are comma separated. Decimal separator in latitude and longitude is a point.

Example file:

```
1
0072, 50.9313, 5.39570, smoking="no"
```

[5] http://kivy.org
[6] kivy: http://www.lfd.uci.edu/%7Egohlke/pythonlibs/#kivy
[7] pygame: http://www.lfd.uci.edu/%7Egohlke/pythonlibs/#pygame
[8] for using kivy on MacOSX see http://kivy.org/docs/installation/installation-macosx.html

2.2.2 Configuration file

The configuration file is a plain text file. In this file one or more blocks starting with '[Indoor]', '[Outdoor]' or '[Both]' are found. After these headlines the names of the SURs follow that belongs to these categories of area of influence.

Example file:

```
[Indoor]
access:age="21+"
camera="no"
[Outdoor]
fishing="no"
littering="no"
[Both]
access:dog="no"
open_fire="no"
```

2.2.3 Keyhole Markup Language

Output files are in .kml format where kml stands for Keyhole Markup Language. For syntax definition and documentation please visit `https://developers.google.com/kml`. For output kml version 2.1 is used.

The GUI version provides also the possibility to read in kml files to display the areas. For this kml version 2.1 as well as 2.2 is supported.

2.3 Command line and GUI version

isySUR comes with two versions: One for command line use and another one for users who prefer to use a graphical user interface. For using the command line interface type:

```
$ run_isySUR.py cli [parameters]
```

For using the graphical user interface type:

```
$ run_isySUR.py
```

Notice that the gui version has more requirements than the command line version. See 2.1.1 for more details.

2.3.1 Command line version

Usage of the command line version:

```
$ run_isySUR.py cli [-h] [-c CONFIG] input output
```

Parameters

The parameters of `run_isySUR.py cli` (bold arguments are required):

- **input** - Path to the input file containing the SURs. See 2.2 for the required format.
- **output** - Path where to put the output file(s). If this path points to a file, a single output .kml-file containing all computed areas is created. If it points to a directory, one .kml-file for each SUR is created in this directory as well as one .kml-file containing all computed areas.
- -c CONFIG (–config CONFIG) - With this parameter an optional config file with SUR classifications (indoor, outdoor, both) is used for computation. For the format of the file see 2.2.
- -h - If this parameter is given, a short help message containing all parameters is shown.

Figure 2.1: GUI with open menu and KML list

2.3.2 GUI version

Usage of the gui version:

```
$ run_isySUR.py gui [-h] [-c CONFIG]
```

If you do not want to give parameters you can also start the gui version with

```
$ run_isySUR.py
```

or a simple double click on the file.

When the gui version could not be started, e.g. because of missing requirements, you can proceed with the command line version. You will be asked for an input file and an output path, to run the program. If you do not want to use this version you can type `'exit'` to exit the program.

Parameters

The parameters of `run_isySUR.py gui`:

- -c CONFIG (–config CONFIG) - With this parameter an optional config file with SUR classifications (indoor, outdoor, both) is used for computation. For the format of the file see 2.2.
- -h - If this parameter is given, a short help message containing all parameters is shown.

The menu

On the left side you can find the menu that holds the tools you will use. In the menu you will find the following possibilities:

- **Load SUR** - Choose the file with the coordinates of your SURs. See section 2.2 for more information about the format. The spheres of influence of the SURs will be calculated. An information text will inform you about the ID of the current SUR. When the kml list is open you can see that spheres are added. Calculated spheres are displayed on the map as well as red markers on the position where the SUR is located. After a calculation is finished you can start another one. The new areas will be added to the ones already in kml list.

- **Load KML** - Choose some kml file you have already saved during the last usage. Or maybe you made your own kml file with another program. See section 2.2 for more information about the format. The areas contained in your file are displayed on the map and added to those already in kml list.
- **Save KML** - All kmls that are currently selected in kml list are saved to a kml file you can choose here.
- **SUR Position** - The red SUR markers can be hidden and shown with this button.
- **Config** - With this button you can open the configuration window. If you used the `-config` parameter there is already a configuration file loaded. You can load another configuration file, add or remove some rules and save your personal configuration file. With your configuration you say if you prefer to find an area for a SUR indoor or outdoor or if you think that kind of rule can be applicable indoor as well as outdoor. The next computations will use the new configuration.

KML list

On the right side there is the kml list. This list tells you which areas you have calculated or loaded. You can scroll the list if it is too long for the window with the mouse wheel or by dragging it with the mouse. When you click on an entry which area is not in viewport the map will jump to that area. When you click on an entry which area is already in viewport you can hide it. Hidden areas are not saved into a kml file. With another click you can make an area visible again.

2.3.3 Using android

Of course calculating SURs at home is nice but in general you would like to know a sphere of influence of a space usage rule standing in front of it. This is possible on your android device (with internet connection). In the archive you can find the isySUR.apk that you can install on your device. Just copy it on the device e.g. via USB and enable the device option to install from unknown sources. Browsing to the .apk and clicking on it starts the installation. After that you will find isySUR in your apps and can use it exactly like the GUI version described in section 2.3.2.

name of file corre

Development
Data usage
Data structure
Evaluation

3. Technical and implementation details

In this chapter the implemented algorithm to calculate the spheres of influence is explained. You will gain an insight into the development and learn some implementation details. After that an evaluation of the achieved results follows.

section?

Code documentation -> pydoc?

3.1 Development

For this project we decided to use Python. First of all, we all wanted to challenge and expand upon our Python knowledge. Furthermore Python is mostly OS independent and can even be ported to android for portable usage which seems to be a good feature here.

This project was created using test driven development on the unittest layer. This does not only allow us to keep track of unwanted side effects of new functionality, but also helps assigning work load to the different members, because it can be easier to write a failing unittest with a bit of information, that is assigned to someone, than it is to explain all the requirements for some new functionality.

isySUR was created in two steps: First the command line version was created and the first versions of the algorithm were implemented. While the algorithm was improved and reworked it was decided to add a GUI because a lightweight tool for displaying the calculated areas was missing. Furthermore it helped in improving the algorithm by enabling rapid testing of changes. On top of that not all people like to use a command line tool.

For the GUI the kivy framework is used. We base the display of our map on mapview from kivy garden[1] which we adapted to suit our needs.

At the beginning of the project the idea of an app came up and with the kivy framework for the GUI the foundations for this was laid. With buildozer[2] a nice tool for packaging an apk file from our project was found.

3.1.1 Data usage

We deliberately do not use the images in our analysis. This is because they do not give us reliable information about the interesting polygons. The variety of possible signs, including

[1] see `https://github.com/kivy-garden/garden.mapview`
[2] see `https://github.com/kivy/buildozer`

totally unknown ones, (and their placement in/on/around the intended area) make it infeasible to perform robust image recognition or classification. Furthermore the only relevant information we think one could extract from images in special cases refers to an inside/outside classification of the rule. However even this classification would not be reliable and robust enough in the general use case.

One might consider using images and compute vision techniques in the future, when the Google StreetView recordings cover virtually everything and these images are available to OSM as well, but until then we do not believe we can extract useful information from the images.

We do use the rule types/names (e.g. access:21="no") to some minimal extend in the way that we pre classified all the rules in the training set to be applicable indoor, outdoor or both, which we use to narrow down the possible OSM elements around the target coordinates. Unknown rules will always get the "both" classification so that we do not ignore relevant information in those cases. We hope to improve our results (or at least our computational costs) by doing so, but unfortunately most rules are applicable in indoor as well as outdoor, so this might only be a minor improvement.

Maybe an image of the pipeline?

not up to date is

Our main focus lies upon the coordinate of the SUR. In a bounding box (currently 100mmight be reduced further) around the given coordinates, we start by searching for the closest OSM relations and ways (we omit nodes for now, since they hardly provide any connectivity information, necessary to be able to construct reasonable polygons). We use a bounding box of limited size because we want to reduce the amount of data that we need to request from OSM and we work on the assumption that these coordinates should always relatively close to the intended area. Unfortunately some coordinates (at least in the test data) are quite far away from the intended area, which can lead to finding unintended polygons that are closer to the coordinates. Furthermore the OSM data is not complete and even some elements from the test data are not present in OSM, which means, that we have no way of ever finding the intended polygon.

add another section with more detailed algorithm information? or expand this one?

3.1.2 Data structure

In order to be able to work freely with all the data that we require when computing suitable polygons for SURs we have introduced several data structures.

- sur: A wrapper object for the SURs. It enables us to handle the file only once at the beginning.
- osmData: Wrapper for the OSM elements Node, Way and Relation. We convert the xml data we receive from OSM to these objects, so that we do not have to work on a xml for computation.
- kmlData: Wrapper objects for a kml and included placemarks. These will be created by our algorithm and can be saved as a kml-xml to a file.

Furthermore, our data structure splits our program into nice modular chunks that can be reused independently in other project if desired.

3.2 Evaluation

Berechnen Sie nun die Geltungsbereiche der Space Usage Rules aus Ihrer Umgebung mit Ihrer Implementierung aus der ersten Runde. Diskutieren Sie die Korrektheit und die Präzision der von Ihnen berechneten Lösungen für die Space Usage Rules des Testdatensatzes und denen aus Ihrer Umgebung. Vergleichen Sie dazu die berechneten mit den intuitiv beabsichtigten Geltungsbereichen.

4. Added SURs

For the competition at least ten new space usage rules should be found. Here our 20 found SURs are presented.

isy0001.jpg:
smoking="no"

intuitive area of
isy0001.jpg

isy0002.jpg:
smoking="no",
access:dog="no"

intuitive area of
isy0002.jpg

---

isy0003.jpg:
access:dog="no"

intuitive area of
isy0003.jpg

isy0004.jpg:
smoking="no"

intuitive area of
isy0004.jpg

isy0005.jpg: smoking="no"

intuitive area of isy0005.jpg

isy0006.jpg: smoking="no"

intuitive area of isy0006.jpg

isy0007.jpg: smoking="no"

intuitive area of isy0007.jpg

isy0008.jpg: smoking="no"

intuitive area of isy0008.jpg

isy0009.jpg: smoking="no", access:bicycle="no"

intuitive area of isy0009.jpg

isy0010.jpg: access:bicycle="no"

intuitive area of isy0010.jpg

isy0011.jpg: access="no"

intuitive area of isy0011.jpg

isy0012.jpg: access:pram="no"

intuitive area of isy0012.jpg

isy0013.jpg: smoking="no"

intuitive area of isy0013.jpg

isy0014.jpg: smoking="no", noise="no"

intuitive area of isy0014.jpg

isy0015.jpg: smoking="no", access:dog="no"

intuitive area of isy0015.jpg

isy0016.jpg: access:fire_case="no"

intuitive area of isy0016.jpg

isy0017.jpg: dog_waste="no"

intuitive area of isy0017.jpg

isy0018.jpg: smoking="no"

intuitive area of isy0018.jpg

isy0019.jpg: access:bicycle="no"

intuitive area of isy0019.jpg

isy0020.jpg: access:dog="no", access:skate="no"

intuitive area of isy0020.jpg

Todo list

add pydoc file 6
name of file correct? 13
section? 15
Code documentation -> pydoc? 15
Maybe an image of the pipeline? 16
this is not up to date is it? 16
add another section with more detailed algorithm information? or expand this one? 16
Berechnen Sie nun die Geltungsbereiche der Space Usage Rules aus Ihrer Umgebung mit Ihrer Implementierung aus der ersten Runde. Diskutieren Sie die Korrektheit und die Präzision der von Ihnen berechneten Lösungen für die Space Usage Rules des Testdatensatzes und denen aus Ihrer Umgebung. Vergleichen Sie dazu die berechneten mit den intuitiv beabsichtigten Geltungsbereichen. 16