Micro-tasking, an efficient method for data acquisition for digital maps?

Anne Sofie Strand Erichsen

December 7, 2016

Contents

List of Figures

Abstract

TB!

1 Introduction

Voluntary acquisition of data for digital mapping has become quite popular over the last years. This is also a cornerstone for OpenStreetMap (OSM), which is a global open access database for digital maps. To establish data sets (for maps) is a comprehensive task. By dividing the work into many small sub-tasks, each of these tasks may be possible to handle by voluntary enthusiasts. This is called micro-tasking.

The Norwegian government has indicated that the most detailed map data for Norway (FKB) may become freely available within a few years. If so, these data may be included in the database of OSM. This paper will look at the FKB building datasets.

An import of FKB building data could result in a 3D modell of the entire country, free and available for everyone. FKB contain very detailed building data and gives a good foundation for 3D modelling of buildings. Trondheim will be a test municipality in this paper. Here buildings are today added manually by users drawing over aerial photos. Height and roof shape is not included, making 3D modelling of buildings very difficult. An example of 3D modeling of buildings in Trondheim added manually from aerial photo is shown in figure 1.1.

Figure 1.1: 3D modeling of buildings in Trondheim, source: osmbuildings.org

The same buildings 3D modeled with FKB building data is shown in figure 1.2. This model is created by Norkart.

Figure 1.2: 3D modeling of buildings in Trondheim, source: 3d.kommunekart.com

Importing data into OSM can be tricky and finding the most optimal approach is not easy. An import should always follow the OSM import guidelines. In OSM imports must be planned and executed with more care and sensitivity than other edits. A bad import can have significant impacts on both existing data and the local mapping

community. Finding a balance between manual verification and automatic import is the key. An import should not demand more time than manually drawing buildings into OSM. When importing data into OSM the team needs a plan on how to The micro-tasking method may be a solution. Two big building imports has used this method and this paper will look at how they did it, and how it worked and what didn't work.

During this paper the student is suppose to:
- Study relevant literature
- Study how Microtasking is working
- Explore how data from FKB can be mapped into OSM
- Investigate if Microtasking is a possible method for including FKB in OSM

2 | Motivation

There are two types of people in the free and open source software for Geoinformatics community: Those who have an geomatics background and learned ICT afterwards and those who have an ICT background and learned geomatics on their own afterwards. The last type has a much higher percentage in the OSM community.

3 | Characteristics of Open Street Map

3.1 General

The OpenStreetMap project is one of the most impressive projects of Volunteered Geographic Information on the Internet[Neis and Zipf, 2012]. Until recent the mapping of the Earth was preserved highly skilled, well-equipped and organized individuals and groups. One important happening was in 2000 when Bill Clinton removed the selective availability of the GPS signal **??**. This change improved the accuracy of simpler, cheaper GPS receivers so that also ordinary people could start mapping their movements. OpenStreetMap was founded in 2004 at University College London by Steve Coast. The goal was to create a free database with geographic information of the world [Neis and Zipf, 2012]. Back in 2004, the geographic data was expensive and hard to get access to.

The OSM project stands out from other data sources mainly because it's free to use and released under a license that allows for pretty much whatever the user wants to as long as the user mention the original creator and the licence[Chilton et al., 2009]. The most common contribution approach is to record data using a GPS receiver and edit the data using one of the free and available OSM editors [Neis and Zipf, 2012].

One of the reasons for OpenStreetMap's success is that today the world has a need for instant information, particularly in crisis situations [Chilton et al., 2009]. Here OpenStreetMap is the leading global example of the effectiveness of crowdsourcing of geodata. Crowdsourced geographic data has characteristics or advantages of large data volume, high currency, large quantity of information and low cost [Wang et al., 2013]. The project is changing the way individuals and organizations are thinking about the collection process, purchase and use of geodata [Chilton et al., 2009].

3.2 Data structure

OpenStreetMap uses a topological data structure. This structure includes three basic components nodes, ways, and relations. Nodes are points with a geographic position stored as coordinates (Lat, long) according to WGS84. Ways are lists of two or more nodes, representing an open- or closed way used to describe streets, rivers, among others [Debruyne et al., 2015]. A relation is a multi-purpose data structure that documents a relation between two or more components [community OpenStreetMap,]. OpenStreetMap's structure uses tags to add metadata to geographic objects. Tags consist of two items, a key and a value of the form key=value. The key is used to describe the topic, category or type of feature, while the value represents the details of the particular form of the key specified. An example of a key-value pair can be building=church, here the key is building and the value is a church, this is a building that was built as a church.

OpenStreetMap does not have any restrictions on tags assigned to nodes, ways or relations, and mappers can use any key-value pair in their import. Nevertheless, the norm in OSM is to try to map new data with existing tags. Good practice is to search for tags, or map features, on different OSM wiki-sites. On the *tags you like* wiki page they recommend different sites, but points out *taginfo.openstreetmap.org* as the most useful site. Taginfo is a website created for finding and aggregating information about OSM tags, it covers the whole planet and is updated daily. The web page list tags used in the database and also inform on how often they have been used.* Taginfo also lists other tags which have been used in combination with the displayed tag. Some countries also have their own taginfo web pages, like Ireland, Great-Britain, and France, Norway does not have it. If a mapper doesn't find an appropriate key-value pair and wants to create a new feature, this has to be documented on the OSM wiki page.

In OSM, changes made by one user over a short period is called a changeset [OpenStreetMap, c]. Change can be a creation of new components, adding tags to existing components, changes to tags in existing components, removal of tags and removal of components. Changes are added to the changeset as long as it's open, changesets are either closed directly or by itself after a period of inactivity (currently after one hour). Every component in the OSM database is a part of a changeset.*

3.3 Organization

The OpenStreetMap Project is supported by the OpenStreeMap Foundation (OSMF) which is a UK-registered non-profit organization. OSMF was founded in 2006 and consists of members from all over the world, as of December 2015 consist of 350 normal-, 351 associate- and 18 corporate members [OSMF, 2015]. OSMF include a board of seven members and is critical to the ongoing function and growth of the OpenStreetMap project [OSMF,]. The foundation has the responsibility for the servers and services necessary for hosting the OSM project. They also support and communicates with the working groups, and delegates important tasks like web page development, etc.

A person can contribute to the OSM project without being a member of the foundation. The project has over 3 million registered users, and around 40 000 active users each month [OSMstats, 2016] collecting, updating and editing the data. The crowdsourced data are then released under the Open Database License, *"a license agreement intended to allow users to freely share, modify, and use this Database while maintaining this same freedom for others"* [ODbL,]. Users can edit maps through different tools made by OSM developers. One tool is called iD and is the default web browser editor created by MapBox. There are also desktop editing applications like JOSM and Merkaartor which are more powerful and better suited for advanced users.

Communication is done through channels like mailing lists, wiki pages, conferences and GitHub repositories. In the public mailing lists, everyone who subscribes to it is overhearing every conversation. Overhearing conversations through mailing lists is described as broadcast communication in the Gutwin paper from 2004 [Gutwin et al., 2004]. The ability to speak to an expected audience rather than one individual has several advantages. Allowing people to decide for themselves whether to respond or not and as the conversation develops new people can join. In OpenStreetMap there are over 150 mailing lists [Reiter and Barroso, 2016], keeping an overview of everything is impossible. That is why weeklyOSM was created in 2010 and is a collection of news relevant to the OSM community written in 5 different languages [Freyfogle, 2016]. In State of the Map 2016 conference, the WeeklyOSM team won the Influential Writing Award, nominated and voted by the community [OpenStreetMap, 2016a]. A good evidence of how important their work is to the community. State of the Map is the main OSM conference, organized by OSMF and has been held each year since 2007 [OpenStreetMap, i]. Important communication is also done through both issues and pull requests in the repositories at the OpenStreetMap GitHub channel.

3.4 File format, .osm files

The .osm file format is specific to OpenStreetMap, and it is not easy to open these files using GIS-software like QGIS. The file format is designed to be easily sent and received across the Internet in a standard format. Therefore .osm files are easily obtained, but using the files directly for analyzing and map design is not easy. The .osm files are coded in the XML format. It is recommended to convert the data into other formats when using the files [OpenStreetMap, d].

Points are represented as nodes, lines as ways and areas as a relation in .osm files. Each represented by a tag: <node>, <way> and <relation>. Node is one of the core elements in the OSM data model. A node-tag consists of a single point defined by node-id, latitude, and longitude. When nodes are used on their own, which means that they are not included in a way or a relation, they represent point features. Points features normally include at least one tag to define the points purpose [OpenStreetMap, f]. In listing 3.1 the node tag describes a bag shop named Citybag, this is called a point of interest. The user key's value is the name of who last modified this node (user="Peter Bremer") and uid are the person's numeric user id (uid="366321"). *

Listing 3.1: Example of a node tag

```
<node id="4004323486" visible="true" version="1" changeset="37189343"
  timestamp="2016-02-13T15:58:54Z" user="Peter␣Bremer" uid="366321"
  lat="63.4318129" lon="10.3971411">
  <tag k="name" v="Citybag"/>
  <tag k="shop" v="bag"/>
</node>
```

A way-tag consists of two or more nodes and can either be open or closed. An open way describes a line feature that does not share the first and last node. Examples of features are roads, cycleway, and streams. When a way is closed the first and last nodes are the same and can be interpreted as a closed polyline or an area, or both [OpenStreetMap, j]. A closed way with highway=* tag can represent roundabouts, or if it has amenity=school tag the closed way represent the outline of a school. In listing 3.2 the way describes a building outline since the key equals building and the value equals church. The building in listing 3.2 is the footprint of a church. The <nd> tag represents a node, where the <nd> tags refers to <node> tags who contains the lat, long values. All <nd> tags create the building footprint, notice that

there is no height parameter. Listing 3.2 creates a 2D representation of the church which is shown in figure 3.1. A 3D representation of the same church is shown in figure 3.2.

Listing 3.2: Example of a way tag - creating the building footprint of a church

```
<way id="89340594" visible="true" version="6" changeset="42571811"
 timestamp="2016-10-01T22:11:17Z" user="Peter Bremer" uid="366321">
 <nd ref="1036369169"/>
 <nd ref="1036369134"/>
 <nd ref="1036369111"/>
 <nd ref="1036369185"/>
 <nd ref="1036369163"/>
 <nd ref="1036369118"/>
 <nd ref="1036369099"/>
 <nd ref="4427055078"/>
 <nd ref="1036369179"/>
 <nd ref="1036369158"/>
 <nd ref="4427055082"/>
 <nd ref="1036369145"/>
 <nd ref="1036369124"/>
 <nd ref="1036369103"/>
 <nd ref="4215548739"/>
 <nd ref="4215548736"/>
 <nd ref="4215548737"/>
 <nd ref="4215548740"/>
 <nd ref="1036369182"/>
 <nd ref="1036369140"/>
 <nd ref="1036369115"/>
 <nd ref="1036369096"/>
 <nd ref="1036369135"/>
 <nd ref="1036369149"/>
 <nd ref="4427055803"/>
 <nd ref="1036369131"/>
 <nd ref="1036369107"/>
 <nd ref="1036369169"/>
 <tag k="amenity" v="place_of_worship"/>
 <tag k="building" v="church"/>
 <tag k="denomination" v="protestant"/>
```

```
  <tag k="name" v="Vår Frue kirke"/>
  <tag k="religion" v="christian"/>
  <tag k="wheelchair" v="yes"/>
  <tag k="wikidata" v="Q3356455"/>
  <tag k="wikipedia" v="en:Vår Frue Church"/>
</way>
```

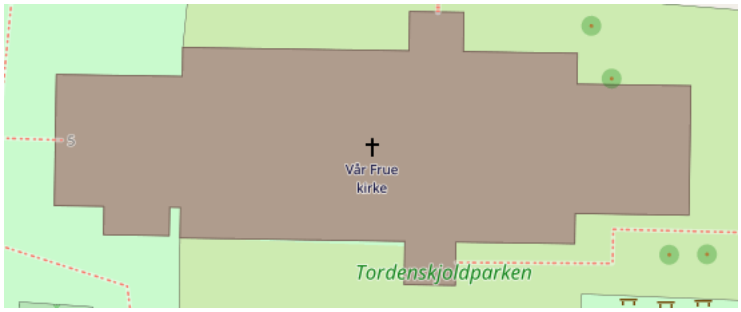

Figure 3.1: 2D representation of *Vaar frues kirke*, result of listing 3.2. Source: openstreetmap.org

A relation is an ordered list of one or more nodes, ways, and/or relations and is used to define logical or geographical relationships between the other elements [OpenStreetMap, h]. If a building consists of multiple parts, tagged with building:part=*, a relation is used to define the geographical relationship between the parts. Specifying roles to different parts is possible. A road can have role as east, going towards east. In multi-polygons, parts can have an inner or outer role, to specify whether it forms the inner or outer part of the polygon. A building relation is shown in listing 3.3. There are eight members in this relation. The first member is a way, and it has an outline role creating the building footprint, for 2D representation. Listing 3.2 is the XML-code for this way (notice that the ref number are equal to the way id in listing 3.2). The only node member in the relation contains the church's address. The rest of the members creates the 3D representation of the building shown in figure 3.2.

Listing 3.3: Example of a relation tag - creating 3D representation of a church

```
<relation id="6269954" visible="true" version="2" changeset="39708156"
  timestamp="2016-06-01T11:14:18Z" user="Peter Bremer" uid="366321">
  <member type="way" ref="89340594" role="outline"/>
  <member type="node" ref="2957446972" role=""/>
```

```
  <member type="way" ref="421821942" role=""/>
  <member type="way" ref="421821938" role=""/>
  <member type="way" ref="421821939" role=""/>
  <member type="way" ref="421821940" role=""/>
  <member type="way" ref="421821937" role=""/>
  <member type="way" ref="421821941" role=""/>
  <tag k="name" v="Vår Frue kirke"/>
  <tag k="type" v="building"/>
</relation>
```

Figure 3.2: 3D representation of *Vaar frues kirke*, result of listing 3.3. Source: osmbuildings.org

3.5 Mapping buildings in OSM

A building can be represented by nodes, ways or relations in OpenStreetMap. When importing buildings into OSM, the XML-code representing the building must be tagged with Building=*. Frequent occurring values are house, residential and garage, describing the buildings particular usage. Using the building tag in node representations is tolerated but not recommended. A building is much better expressed by their footprint (close way or multi-polygon), and if the footprint is available, one should not add the building key in nodes. The building key should be used to mark the buildings footprint. The most common occurring value for the building key is yes and

used when it's not possible to determine a more accurate value [OpenStreetMap, b]. A list of possible values that can be added to the building key is listed on the OSM building wiki page. It is possible to introduce new values, but it is not recommended. The building key is most common used in way representations [TagInfo, 2016]. An example of how to use building key in a way-tag see listing 3.2.

A building can also be represented by a relation. Relations are used if the building consists of multiple parts which physical differ from each other, often when a 3D representation of the building is created. A building relation mainly consists of two or more ways. A way then represents a part of the building and should be tagged with a building:part key and usually the value yes. Then the way-tags representing the different building parts are ordered together inside the relation. An example of a building:part=yes implementation can be seen in listing 3.4. Note that the first and last <nd> tag refers to the same node, so this is a closed way. The key roof:shape with value gable gives the appearance of the roof. The result from the code in listing 3.4 is shown in figure 3.3 marked with a red line. A relation containing building:parts are shown in listing 3.3. Notice that this relation is tagged with the key type and the value building. If a relation is tagged with type=building, it groups both building footprint and all building parts together. See figure 3.2 for a 3D building representation created with building parts.

Listing 3.4: Example of a way tag - creating 3D representation of a building part

```
<way id="17533469" visible="true" version="22" changeset="39301425"
timestamp="2016-05-13T21:20:04Z" user="Peter Bremer" uid="366321">
<nd ref="3505716655"/>
<nd ref="2517225923"/>
<nd ref="4184346715"/>
<nd ref="3505716656"/>
<nd ref="4184346713"/>
<nd ref="4184346717"/>
<nd ref="3505716654"/>
<nd ref="4184346719"/>
<nd ref="3505716655"/>
<tag k="building:colour" v="#c3c0b9"/>
<tag k="building:material" v="stone"/>
<tag k="building:part" v="yes"/>
<tag k="height" v="11"/>
<tag k="name" v="Sydkapell"/>
<tag k="roof:height" v="4"/>
```

```
  <tag k="roof:material" v="copper"/>
  <tag k="roof:orientation" v="across"/>
  <tag k="roof:shape" v="gabled"/>
</way>
```

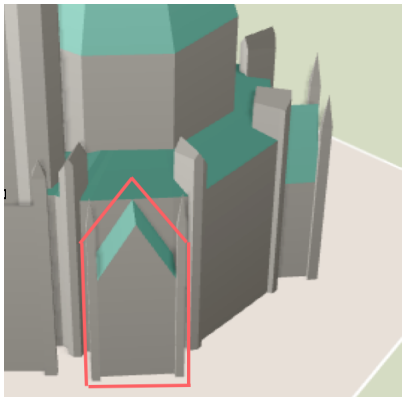

Figure 3.3: *Sydkapell* with gabled shaped roof, result of listing 3.4. *Source: osmbuildings.org*

4 | SOSI and FKB building

This paper will look at FKB buildings and how it can be imported into OpenStreetMap. The FKB dataset comes in SOSI format. But what is SOSI? This chapter will give an introduction to SOSI, both standard, and format. The standard was developed by the Norwegian mapping authority and is the largest national standard for geographic information, where the standard is implemented in the SOSI format. SOSI is a widely used file format for Norwegian mapping [Kartverket, b]. FKB is a collection of datasets containing detailed vector data of Norway, created in the SOSI format. Since this paper looks at FKB building dataset this chapter will also describe FKB and the most common and valuable features for OpenStreetMap within the building dataset. Some features will not be relevant for import into OpenStreetMap. The chapter ends with an evaluation of SOSI, looking at its advantages against OpenStreetMap XML format.

4.1 SOSI

The national standard for geodata in Norway is called SOSI, created by the Norwegian mapping authority. Geodata is map data stored in a digital format so that we can produce maps from it. The standard is based on international standards, primarily the NS-EN ISO19100-family of standards [Skogseth and Norberg, 2014]. The SOSI standard is implemented in the SOSI format, a Norwegian geospatial vector data format. The SOSI data consists of point -, line (*kurve*) - and area (*flate*) features. A point feature is only one single vertex, given in north and east coordinates with or without the height. Line feature is two or more vertices, but the first and last vertex are not equal. An area feature is three or more vertices, where the first and last vertex are equal.

The Norwegian map authority has established a general feature directory *(objekt katalog)* in connection with the SOSI standard. The purpose of a feature directory is to specify feature types and associated properties that are general within a discipline or across multiple disciplines [Kartverket, 2016b]. The directory covers around 50 disciplines (2014) [Skogseth and Norberg, 2014]. In SOSI version 4 the modeling method was changed, from modeling point, line and area to modeling feature types in the real world (buildings, roads, boundaries etc.) [Skogseth and Norberg, 2014]. For instance, a road will have many associated properties, in addition, it can be located as a line, this is how SOSI version 4 models its data.*

SOSI data can be presented on four different levels, each level represents a different data quality. A SOSI dataset contains information on three levels. First are the *Head* containing shared information about the dataset, this information applies to all data. Then comes the *Data itself* containing properties and location coordinates (N, E and height). The SOSI dataset is finished with an *End* which is the end of the data series [Skogseth and Norberg, 2014].

4.2 FKB and data quality

FKB, *Felles Kartdatabase* in Norwegian, is a collection of structured datasets that contains the most detailed vector data of Norway. FKB data is collected through a collaboration called Geovekst. This is a collaboration between the Norwegian mapping authority, the Norwegian road authority, Telenor, Energy Norway, the Norwegian Association of Local and Regional Authorities, the ministry of Agriculture and the Norwegian Water Resources and Energy Directorate. [Kartverket, a]. FKB data comes as vector data in SOSI format [Kartverket, 2011].

The FKB standard describes which features that is included in the mapping and the accuracy of the objects. There are specified four FKB standards, FKB-A, FKB-B, FKB-C and FKB-D [Kartverket, 2011]. FKB-A is the most detailed, containing good three-dimensional data description and has high standards on accuracy (5-20 cm) and content. Most commonly used in city centers. FKB-B is also detailed with an accuracy of 20 - 30 cm. Most commonly used in urban areas. FKB-C is used for overview planning and management (*forvaltning*) with an accuracy of 0.50 - 2.00 m. FKB-D are areas not covered by the three other standards, like mountain areas, and has a broad accuracy of 5 - 100 m. Today, maps should always be produced after the FKB standards [Skogseth and Norberg, 2014].

This paper will look into the FBK building dataset. The data consist of both point, line and area and contains 24 different feature types [Kartverket, 2013a]. The data is established and kept up to date by using photogrammetry. In some cases, the data is * established by using land surveying [Kartverket, 2013b]. Building points are transferred from the cadastre (*Matrikkelen*). Data is* delivered in the official reference system for each municipality since the data are distributed per* municipality.

Today FKB data is saved piecewise within each municipality. The data is collected in a database at the Norwegian mapping authority which is only updated one or two times a year. This is not the optimal solution. A goal is to gather all FKB data, from every municipality, into one central database where all updates will be made directly to this central database. The goal is to have 80% of Norwegian municipalities connected to the database within 2018 [Kartverket, 2016a].

Possibilities for 3D representation of buildings from the FKB standard varies. Some feature types in the FKB dataset have a level of detail attribute called *TREDNIVÅ* where they usually use six levels. At level 0 solely 2D is supported, limited to the ground floor. In level 1, buildings are represented as blocks with a flat roof. The height of the roof is either the minimum, maximum or average of ceiling height around the building. Recognizability is not great. In level 2 the main shape of the roof is maintained with the use of ridge lines and break lines. Photogrammetric data capture for FKB-A, -B and -C standards provides buildings with level of detail similar level 2. Level 3 includes added features as dormers, balconies, larger chimneys etc. Gives a better visual quality and a more appropriate basis for analyses. Photogrammetric data capture for FKB-A and -B standards provides a level of detail similar level 3, but details are different for the two standards. Level 4 is a high-quality model of buildings, not supported in the SOSI standard building model [Kartverket, d]. Level 5 is a high-quality model of a building both outside and inside, not supported in the SOSI standard building model [Kartverket, d]. FKB-A and FKB-B features describing the main roof has to be at least level 2 of detail and features describing details located at the roof with at least level 3 of detail.

4.3 Features in FKB building

Looking at a FKB building dataset covering Trondheim municipality gives some indication of the most common feature types and help to determine which features

should be prioritized in the conversion between FKB building SOSI format and OSM file format. This section will use the Trondheim municipality dataset. The dataset contains 1499 point objects, 618 710 line objects and 95 071 area objects. This municipality has a fairly large city center and a good variation in building types. Therefore this dataset is a good representative of the average municipality in Norway.

There are 24 different feature types in the FKB building dataset. Where two features are point data, three features are area (*flate*) data and the rest is line (*kurve*) data. The features are grouped into four categories, building and building refinement (*bygningsavgrensning*), descriptive building lines, building appendage (*bygningsvedheng*) and lastly roof covering (*takoverbygg*).

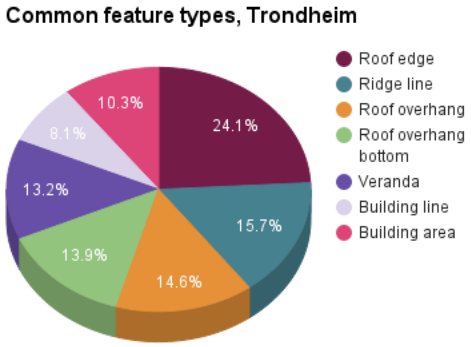

Figure 4.1: Most common feature types in Trondheim

When considering an import to OpenStreetMap there will be features that are not as relevant. Adding all buildings as point data does not seem relevant. The two feature types within point data are building and assistantpoint3D (*hjelpepunkt3D*). The building feature comes as both point and area, containing exactly the same attributes. If the building footprint is available, points should not be imported, as mentioned in section 3.5. Therefore, the building feature as point data should not be imported into OSM, it will not add additional information. The assistantpoint3D will not be imported when excluding the building point feature. Conclusion is that no point data will be relevant for import.*

In Trondheim there are 158 917 roof edge (*takkant*) features and is the most common line feature in this municipality. This feature is the building's exterior roof surface refinement (*avgrensning*). See figure 4.2 and 4.3 for examples on where this feature is mapped on buildings. The second most common line feature is ridge line (*Monelinje*). There are 103 488 objects with this feature in Trondheim. Ridgeline is the line describing the horizontal bending line / break line (*knekklinje*) on top of the roof and is also the highest peak of the roof. See figure 4.2 and 4.3 for example of where it is mapped on buildings. A minimum goal for the FKB mapping team is to map ridge lines on every building [Kartverket, 2013b] and can explain the hight frequency of this feature [Kartverket, 2013b].

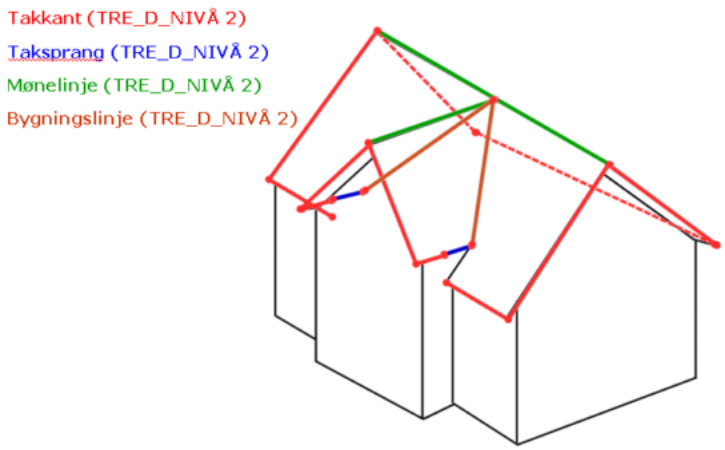

Figure 4.2: Roof edge (red), roof overhang (blue), ridge line (green) and building line (brown) [Kartverket, 2013b]

Third most common line feature in Trondheim is roof overhang (*taksprang*) with 96 436 objects. This feature describes the top of the roof edge inside the building shell, not located on the outside edge which is the roof edge feature. For an example see figure 4.2 and 4.3. This feature should be mapped where height difference between two roof levels is larger than the tolerance of the FKB-data. This feature is in the descriptive building lines category. The line always follows the roof edge.

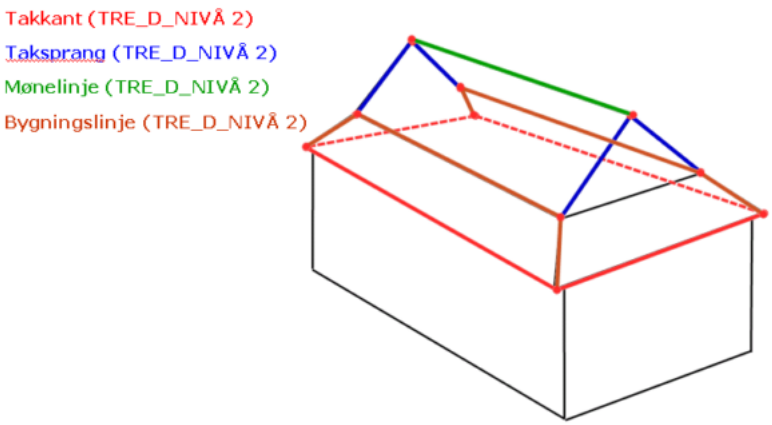

Figure 4.3: Roof edge (red), roof overhang (blue), ridge line (green) and building line (brown) [Kartverket, 2013b]

Fourth most common line feature in Trondheim is roof overhang bottom (*TakSprangBunn*) with 91 281 rows. This feature describes lines located at the bottom of a roof edge within a building mass (*bygningskropp*). Roof overhang bottom is under the descriptive building lines category. In figure 4.4 the blue line shows where a roof overhang bottom line can be drawn. As shown in figure 4.4 roof overhang bottom lines should, if possible, have equal coordinate values (N, E) as the corresponding roof overhang. This is visualized with the dashed black lines on figure 4.4. In figure4.4 the red and blue lines are from the original figure in [Kartverket, 2013b], the green line is added afterwards to visualize roof overhang in the same figure.

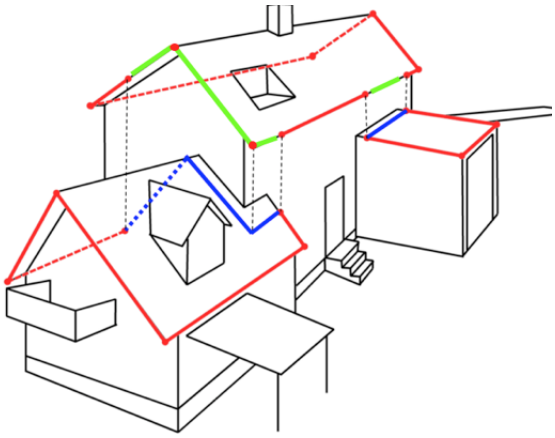

Figure 4.4: Roof overhang bottom (blue), roof edge (red) and roof overhang (green)

Fifth most common line feature in Trondheim is veranda and includes veranda, terrace, balcony and loading ramp [Kartverket, e]. If the area mapped is following FKB-A standard veranda features down to 2 square meters are drawn, and down to 6 square meters if the area is following FKB-B standard. Veranda features have an attribute value MEDIUM that describes if it is located on the roof (MEDIUM = B), on the outer wall (MEDIUM = L) or on terrain (MEDIUM = T). This attribute is helpful when making 3D models of the buildings. Height attributes can either be a reference at the top of the railing (used for medium B) or at floor level (used for medium T). When the feature has attribute medium L its optional which height reference to use. See figure 4.5 for example of veranda features. Veranda features are under the building appendage category.

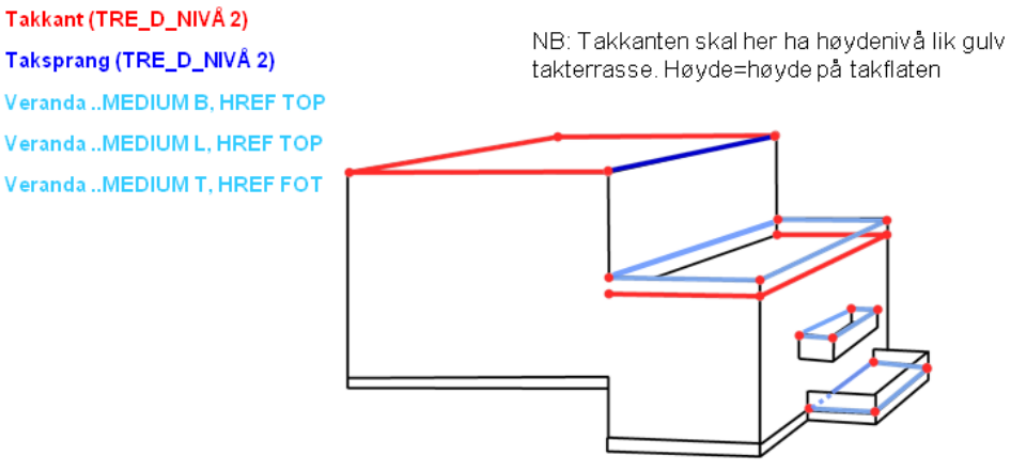

Figure 4.5: Roof edge (red), roof overhang (blue) and veranda (turquoise) [Kartverket, 2013b]

Sixth most common line feature in Trondheim is building line (*bygningslinje*) with 53 255 rows. The feature describes building details which are within a roof perimeter, and which cannot be described by other feature types. If data covering the area is within the FKB-A standard the building lines should be drawn on objects that are 2 cubic meters or larger and if it's within the FKB-B standard only objects that are 7.5 cubic meters or bigger should be drawn. The two limits are just instructive, some interpretations will be done. Examples of use can be seen on figure 4.2 and 4.3.

Most important area feature is building, the two other features are otherBuilding and RoofCovering/RoofSuperstructure, like a carport. There are three types of

building demarcation (*bygningsavgrensning*) defined in FKB, foundation wall outline (*grunnmurriss*), facade outline (*fasaderiss*) and roof outline (*takriss*. If more than one of them exist, roof outline will form the surface of the building feature. Building feature has an type of building attribute, which is an initial value mapped to a specific building type. House (*enebolig*) has initial value 111 and dormitory (*studenthjem*) value 152 [Kartverket, c]. Height reference is either measured from top, bottom or is unknown.

It is impossible, at least in FKB, to create a specification of registration of buildings that are completely accurate. The buildings will always be subject to some generalization. This can be seen in the figures used in this section.

4.4 Benefits using SOSI in context of OSM

Listing 4.1 is SOSI code for area representation of a building. FLATE means area and contains the feature id. OBJTYPE is feature type and in listing 4.1 equals building, representing what the key would be in OSM. KOMM field contains a municipality code, representing the buildings municipality. BYGGNR is the buildings unique number found in the cadastre. BYGGTYP NBR is the description of what the building actually is used as or approved as. For instance in listing 4.1 BYGGTYP NBR equals 182, meaning this building is a garage attached to a holiday home (*fritidsbolig*) [SOSI-sekretariatet,]. This attribute will give the value to the building key in the mapping to OSM. BYGGSTAT is the building status where value TB means that the building is being used.

Area representations in SOSI can refer to other geometry types. This is similar to how way and relation representations can refer to other objects in OSM. The REF field refers to the other geometry features using the feature id for the object being referred to. In listing 4.1 REF equals -166775, which refers to the line feature in listing 4.2. The minus sign in REF's value means that it refers to the line in listing 4.2 but with opposite direction.

Listing 4.1: Example of a area representation of a building in SOSI

```
.FLATE 715280:
..OBJTYPE Bygning
..KOMM 1601
..BYGGNR 196122907
..BYGGTYP_NBR 182
```

```
..BYGGSTAT TB
..KOPIDATA
...OMRÅDEID 1601
...ORIGINALDATAVERT "Trondheim kommune"
...KOPIDATO 20160502
..REF :-166775
..NØ
703226400 55444400
```

Listing 4.2 is a line representation of a building part. Here the feature type is roof edge, examples of a roof edge is shown in figure 4.2 and 4.3. OSM XML code of the same roof edge is shown in listing **??**. Both listings refer to 6 different point representations. In listing 4.2 the point coordinates are written directly in the representation. NØH is north, east and height value for each point. KP1 means that this point is a connection point, but this is not implemented in the OSM XML code. In listing 4.3 the XML codes refers to 6 different nodes (first and last node are the same). This is not possible in line representation through SOSI format. Only area representation can refer to other geometry features [Kartverket, 2006]. This is one major difference between SOSI format and OSM XML format.

Listing 4.2: Example of a line representation of a building part in SOSI

```
.KURVE 166775:
..OBJTYPE Takkant
..DATAFANGSTDATO 20100610
..VERIFISERINGSDATO 20150627
..REGISTRERINGSVERSJON "FKB" "4.01"
..KVALITET 24 19 0 24 23
..TRE_D_NIVÅ 2
..KOPIDATA
...OMRÅDEID 1601
...ORIGINALDATAVERT "Trondheim kommune"
...KOPIDATO 20160502
..NØH
703226612 55444485 1344 ...KP 1
..NØH
703226525 55444618 1280
703226160 55444380 1280
703226247 55444247 1344 ...KP 1
```

```
..NØH
703226328 55444123 1283
703226693 55444361 1280
703226612 55444485 1344 ...KP 1
```

Listing 4.3: Example of a line representation of a building part in OSM XML

```
<way id="-166775" version="1" visible="true">
        <tag k="KOPIDATO" v="20160502"/>
        <tag k="OMRÅDEID" v="1601"/>
        <tag k="KVALITET" v="24; 19; 0; 24; 23"/>
        <tag k="TRE_D_NIVÅ" v="2"/>
        <tag k="ORIGINALDATAVERT" v="Trondheim kommune"/>
        <tag k="REGISTRERINGSVERSJON" v="FKB; 4.01"/>
        <tag k="OBJTYPE" v="Takkant"/>
        <tag k="VERIFISERINGSDATO" v="20150627"/>
        <tag k="DATAFANGSTDATO" v="20100610"/>
        <tag k="KURVE" v="166775"/>
        <nd ref="-161600" />
        <nd ref="-387568" />
        <nd ref="-387569" />
        <nd ref="-387570" />
        <nd ref="-387571" />
        <nd ref="-387572" />
        <nd ref="-161600" />
</way>
```

Listing 4.4 is the point features the way XML code listing 4.3 refers to.

Listing 4.4: Example of a line representation of a building part in OSM XML

```
        <node id="-161600" lat="63.4147679" lon="10.0904069"
              version="1" visible="true"/>
        <node id="-387568" lat="63.4147599" lon="10.0904333"
              version="1" visible="true"/>
        <node id="-387569" lat="63.4147275" lon="10.0903844"
              version="1" visible="true"/>
        <node id="-387570" lat="63.4147355" lon="10.0903580"
              version="1" visible="true"/>
        <node id="-387571" lat="63.4147430" lon="10.0903335"
```

```
        version="1" visible="true"/>
<node id="-387572" lat="63.4147754" lon="10.0903824"
        version="1" visible="true"/>
```

The roof edge represented by listing 4.3 is shown in figure 4.6 where the numbers represent the drawing order. The point marked with 1 is *<nd ref="-161600"/>* and the point marked with 6 is *<nd ref="-387572" />*.

Figure 4.6: Roof edge representation

There are multiple similarities between the SOSI format and OSM XML format as shown in this section. Neither SOSI or OSM has a polygon representation directly and both have open and closed line representations. Feature types in SOSI are easily mapped over to key-value relations in OSM. FKB's way of representing buildings is also benefitial when modeling building parts in OSM, especially in 3D modeling. FKB contains line representation of every building part. This is why a SOSI to OSM converter is beneficial instead of a shape to OSM converter for instance.*

5 | Micro-tasking

To be able to import FKB building data into OpenStreetMap a import method is desired. A promising method is micro-tasking. This chapter will give an introduction to what the micro-tasking method is. It will describe how the method was initially introduced in OpenStreetMap, what micro-tasking is and introduce related tools using this method.

5.1 Background

After the Haiti earthquake in 2010 Humanitarian OpenStreetMap (HOT) was formally registered as a non-profit organization [Soden and Palen, 2014]. During a 3-week period 600 remotely located volunteer mappers built a base layer map for Haiti nearly from scratch [Soden and Palen, 2014]. The Haiti mappers were loosely organized through public lists, real-time chat (IRC) discussion and a wiki page [Palen et al., 2015]. One organizer wrote to the OSM mailing list "*What tools are available to see in real time which areas have been mapped recently? [...] This would be helpful for general coordination among us mapping in Haiti*". The request was unfulfilled. In the Palen paper from 2015, they found a case of duplication of two changesets created less than 1 minute apart contained four duplicate road sections created by two mappers. Findings like this one illustrate how high tempo mapping in a limited region can result in collisions.

Solving collisions in the aftermath of the Haiti earthquake led the HOT-team to create the OSM Tasking Manager tool, the first version finished in 2011. This tool where thought as a help to mappers to more efficiently coordinate simultaneous work. This tool reflected the growing popularity of micro-tasking, or a Find-Fix-Verify crowd programming pattern, as a solution to managing and implementing distributed

work [Bernstein et al., 2015].

5.2 Micro-tasking method

The simplest type of tasks are called micro-tasks and should be accomplished from a few minutes to a few hours. The tasks are simple tasks, often highly repetitive. Tasks are often grouped together into one project or a campaign. In Non-profit organizations, it can be hard to get enough people involved, especially if it requires a lot of time from the volunteers. Micro-volunteering helps people volunteer without demanding all their time. The volunteers are only required to work in a limited time completing as many micro-tasks as their available time allows. When using the OSM Tasking Manager on mapping projects volunteers can complete mapping tasks within a reasonable time interval. This is a good solution for getting more volunteers to contribute who struggle to fit the volunteer work into their busy schedules. This concept has a huge potential but lacks awareness [Bernstein et al., 2013]. The term micro-volunteering appears from the Spanish organization "Microvoluntarios", an online platform which allowed charities to post micro-tasks and connect to volunteers who can perform the tasks [Madalena and Clara, 2015]. Strengths of using micro-tasks are, among others, flexibility and convenience for the mappers [Madalena and Clara, 2015]. Jim McAndrew said in his NYC state of the map US 2015 speak that micro-tasking can benefit OpenStreetMap and gives a lot of opportunities to the community, micro-tasking is the way of the future [McAndrew, 2015].

5.3 OSM Tasking Manager

Micro-tasking in OpenStreetMap is done through, among others, the OSM Tasking Manager tool. The purpose of the OSM Tasking Manager is to divide a mapping job into smaller tasks. The tool improves project awareness since information about the project and tasks is very easily accessible. The interface present the user with an overview of which areas needs to be mapped and which areas needs validation. It has an overview of how much work is left, how much is finished and general information about the mapping and introductions on how to do it. A common mapping project can be to map every building in an area using aerial photos. The area is divided into equal grids and displays different color codes. Yellow means that this grid is being mapped by someone else, red means that the area is marked

as finished but are waiting for validation by another user and green indicates that this area has been inspected in the OSM database for completeness and compliance [Palen et al., 2015]. Empty grids mean that the area covered needs mapping. An example of a project in the OSM Tasking Manager is shown in figure 5.1. The project aims at mapping road networks, residential areas and buildings in the area marked [HOT Tasking Manager, 2016].

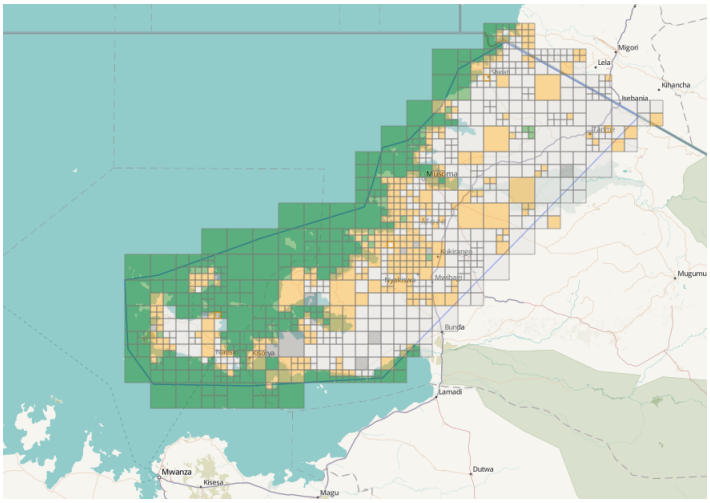

Figure 5.1: Project 2261- Tanzania Development, source: tasks.hotosm.org

The OSM Tasking Manager has in the aftermath developed to other projects outside the HOT community. Especially import projects have had good leverage in this tool, more about this in chapter 6. Newer versions of this tool gave mappers the possibility to asynchronous communicate on tasks, making it easier to inform each other. Unfinished tasks often contain information from users who have been working on it with an explanation of what they mapped or what they didn't map. The Los Angeles building import, which started in 2015, developed the OSM Tasking Manager 2, adding new features to fit their needs. Adding possibilities for tasks being polygons and not grids.

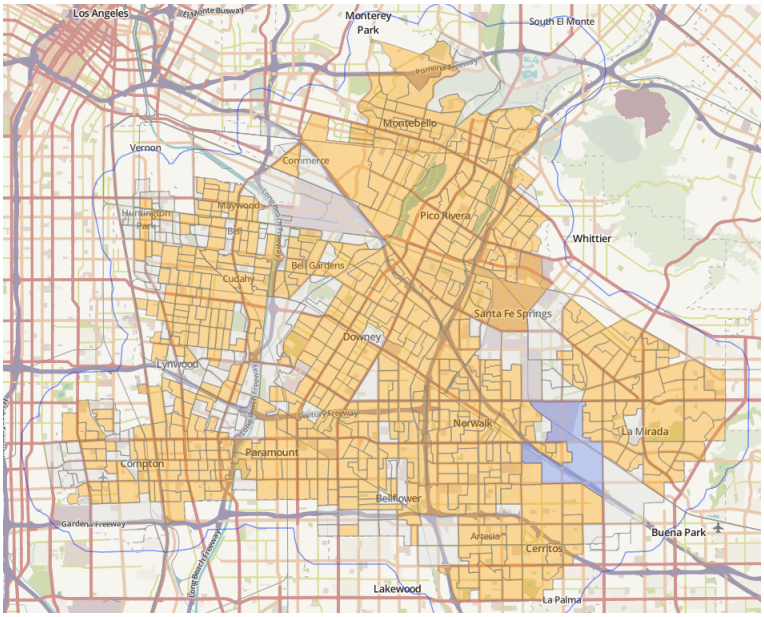

Figure 5.2: Project 2 0- Southeast - LABuildings Import, source: labuildingsimport.com

An example of the OSM Tasking Manager 2 used by the Los Angeles building import team is shown in figure 5.2. Notice that instead of grids they use polygons, dividing the tasks into a reasonable size using census block groups. More about this in subsection 6.3.

5.4 Other tools

Other tools in OSM that also use the micro-tasking method are often designed for fixing bugs in the OSM database. Both Map-roulette and To-fix are examples of such tools and are listed as error detection tools on the OSM quality assurance wiki page. MapRoulette was created by Martijn van Exel and Serge Wroclawski, and its slogan is "*Towards a better map, one bug at a time.*" This tool has a gamified approach to fixing bugs, breaking common OSM data problems into micro-tasks. According to its founder Martijn, MapRoulette has become a very popular mapping pastime for both amateur contributors and experienced mappers [Exel van, 2013]. The tool breaks tasks that need fixing into challenges with a tutorial on how to best

fix the issues in the challenge. It displays one issue at a time, once every issue in a task is fixed a new challenge is activated. Common issues are connectivity errors among ways, overlapping ways and equivalent ways [OpenStreetMap, e]. To-fix is a micro-tasking tool creates by MapBox and is very similar to MapRoulette. This tool breaks up large mapping jobs into smaller tasks that multiple people can work on asynchronously [Lidman, 2014]. The tasks are queued up, then a group of mappers can do the tasks and track their progress. Tasks presented here are also commonly fixing overlapping and crossing highways. With To-fix, issues can be fixed through the iD editor or JOSM editor. This tool do not give an introduction to the tasks like MapRoulette, making it more challenging knowing what to do. To-fix loads tasks much faster then MapRoulette who are quite slow. There are also bugs in MapRoulette. For instance, half of the information text are not displayed because it's too long for the information box and the skip button do not work properly. Both solutions convey statistics presenting total fixed, skipped and marked as not error on tasks.

6 | OSM import methods

The traditional way to contribute data to the OpenStreetMap project is through active users who use their GPS to track roads and their local knowledge to add information about geographic regions to the OSM database [Zielstra et al., 2013]. Users also digitalize aerial photos by drawing buildings etc. from it. Cheaper GPS receivers and more available satellite imagery with better resolution make it easier for users to contribute [Chilton et al., 2009]. The number of active users in different regions varies, making some regions on the OSM map full of data while others are almost empty. The varying number of active mappers led to a second approach for getting data into the OpenStreetMap database; bulk imports [Zielstra et al., 2013]. Bulk import is the process of uploading external data and was meant for initial/preliminary object class uploads, so only if the object were none existing in the area [Zielstra et al., 2013]. It's a good alternative for countries or regions with few or none active users. Through the years different import methods have been developed. This chapter will evaluate the three common import methods. A fully automatic method, a guided automatic method and the micro-tasking method.

6.1 Fully automatic import script

Creating a script that automatically imports big datasets into OpenStreetMap, a bulk import, is not encouraged in the OSM community [Zielstra et al., 2013]. This becomes clear when reading the wiki pages about import. A bulk import is supposed to be a supplement to user generated data. The user-generated data and the user's ability to work is always the priority [OpenStreetMap, l]. A fully automatic import do automated edits to the OpenStreetMap database with little, if any, verification from a human. Automatic edits are changes that have no or very limited human oversight [edits OpenStreetMap,]. This kind of edits must follow the Automated

Edits code of conduct [OpenStreetMap, a]. The policy was created to prevent damaging acts on the database and ignoring it will result in the import be treated as vandalism.

An example of a bulk import was the TIGER import. The Topologically Integrated Geographical Encoding and Reference system (TIGER) data was produced by the US Census Bureau and is a public domain data source. The bulk import was completed in early 2008 [Zielstra et al., 2013], populating the nearly empty map of the United States. The TIGER dataset was not perfect and had its faults, but it was better than no data at all [Willis, 2008]. The import mainly focused on data containing the general road network of the US [Zielstra et al., 2013].

Reading through the OpenStreetMap mailinglist called talk, it becomes apparent that this import is an automatic edit with very limite oversight. "*Please don't upload Northampton County, [..] I've mapped my entire town there [..]*" [Mielczarek, 2007]. Active OSM mappers following the maillinglist are trying to save their manual work. Users not attending the mailing list had limited, if any, possibilities of saving their work through the TIGER import. Mappers importing TIGER data tried not to override existing data. They started on empty states but reading through the "TIGER, which states next?" mailing thread the decisions on which state to import next was solely based on the people attending the conversations. *"I believe I'm the only one out here in Nebraska [..] Feel free to override my edits"* [Bishop, 2007]. The import process did not have any requirements on what they should do with existing data. OSM mappers added the TIGER data county by county, state by state. The OSM user Dave Hansen, one of the most active users during this import, created a text file with his upload queue and published it on his dev openstreetmap site [Hansen, 2007b]. He added states and counties after requests from users [Hansen, 2007c][Hansen, 2007a]. The import team did not have any validation or correction methods or routines. In the aftermath of the import, tagging errors of. For instance, the TIGER data groups residential, local neighborhood roads, rural roads and city streets into one road class, while OSM uses a more refined data schema with various highway tags for multiple road classes [Zielstra et al., 2013].

On the OSM TIGER wiki page, last updated August 2016, they say *it is unlikely that the TIGER data ever will be imported again.* The main reason is the growing US mapping community, and their mapping is often better than the TIGER data. *"Do not worry about getting your work overwritten by new TIGER data. Go map!"* [OpenStreetMap, 2007]. A new bulk import with updated TIGER data can overwrite existing, more precise data. The TIGER data are of variable quality, poor road

alignment is a huge problem and also wrong highway classification. Many hours of volunteer work could just be lost, and this is something the community want to avoid. The bulk import in 2007 got the United States on the OSM-Map and saved the mapping community a lot of time finding road names etc. *"TIGER is a skeleton on which we can build some much better maps"* [Willis, 2007]. On the opposite side the project kept the US mappers away, they were told for years that their work was no longer needed after the TIGER upload was complete. But the presence of TIGER data ended up requiring a lot of volunteer help, fixing errors like the poor road alignments and wrong tagging.

Bulk imports are overall not recommended today but have been helpful as well. In the Netherlands, bulk imports have met little resistance, mainly because the import work are done by dedicated OpenStreetMap mappers who know the OSM import guidelines. Arguments for bulk imports say that a map that already contains some information is easier to work on and can help lower the entry barriers for new contributors. Another argument is that an almost complete map is more attractive for potential users, that again can encourage more use of OSM data in professional terms [Exel van, 2010]. But a huge minus to bulk imports is the data aging since the data being imported usually is a few years old and updating it takes time, often years. The TIGER import was data from 2005, but the import finished in 2008 [Zielstra et al., 2013]. Between the time of first import and update, the community has fixed bugs, added necessary metadata, and the community would not want to lose that data/information.

6.2 Guided automatic import

Fully automatic import of huge amounts of data is discouraged in the OSM community, so another approach is guided automatic import. The OSM community encourages people to import only small amounts of data at a time and only after validation and correcting errors [Mehus, 2014]. This method was applied when the OSM-community in Norway got approval from the Norwegian map authority to import N50 data [Kihle, 2014]. N50 is the official topographic map of Norway, and provide a good data foundation. The import process is described on the OSM Topography import for Norway wiki page. The mapping team imported one municipality at a time. Each municipality dataset was divided up in .15 deg time .15 deg grid changesets, and each changeset contained from five thousand to twenty thousand elements [OpenStreetMap, 2014]. The N50 import was a community import, but especially experienced user was

encouraged to import the data [Mehus, 2014].

The N50 release was good news for Norway, since regions, especially in northern parts of Norway, had limited amounts of data because of few active OSM mappers. This import would then increase the quality of OpenStreetMap in much bigger parts of Norway [Jørgenrud, 2013]. It's a huge dataset, so they had to import it with caution, not everything in the dataset was relevant for OpenStreetMap, this is noted by the OSM user Solhagen [Solhagen, 2015].

The OSM user tibnor preprocessed the data and uploaded to a google drive folder available for everyone. He started on the preprocessing early 2015. Late 2013 the OSM user gnonthgol* created sosi2osm script for conversion between SOSI format and OSM file format, which he informed about in the sosi2osm mail thread 1. This script is the recommended way of converting the N50 SOSI files since it converts directly from SOSI to OSM XML format. In May 2015 tibnor released a JOSM plugin for the N50 import of rivers/streams to make it faster to check and fix directions. The import process was managed through a wiki page. Here users wrote their name and progression, start date and end date of the imported municipality. Elements which needed manual inspection or validation was tagged with "Fix-me" and a description of what to do. They used a python script ("replaceWithOsm.py") to merge N50 data with existing OSM data, adding source=Kartverket N50 tags on the new data. Elements that already exists in OSM, that conflicts with the new data, is marked with FIXME=Merge. Here the user has to search for the conflicting elements and correct the errors manually. Then the data can be uploaded to OSM.

The N50 import was initially stopped in May 2014 by Paul Norman. The Norwegian OSM group started importing the N50 data before consulting with the OSM imports mailing list, which is required. They were also importing the data without the proper approach. The wrong import approach was pointed out by DWG member Paul Norman [Mehus, 2014]. DWG is the data working group, created in 2012, and they are authorized by the OSMF to detect and stop imports that are against the import guidelines [OpenStreetMap, k]. The Data working group reverted the import because of technical problems and errors in the import [Hagen, 2014]. The reset was a step back for the Norwegian OSM team, and they had to start over again. The DWG stopping the import in 2014 was probably for the best. They have much experience with automated imports.

The N50 import has been time-consuming. It started in 2015 and is still not finished, even though most of the municipalities are imported. The import process was carried out according to the import guidelines. Without the DWG group, the import would

probably end up as an automated edit with no proper validation process. The N50 import is one example of how time-consuming this process can be. There are guidelines to follow, a lot of validations to be done.

6.3 Import based on micro-tasking

The N50 import was a good start towards micro-tasking an import of huge amounts of data. They divided every municipality into .15 degree .15 degree grid changesets and imported one changeset at a time. Both the New York building import finished in 2014, and Los Angeles building import, not finished, took this mindset to the next level, creating a Tasking Manager interface particular to this import, among other initiatives. The LA-building team created a custom tasking manager to coordinate the LA County building import [OSM Tasking Manager,], while the NY-team used the original OSM Tasking Manager.

In the Guided automatic import from 6.2 we saw that dividing datasets into smaller parts makes the import easier to, among others, distribute the workload between experienced users. OSM Tasking Manager takes this approach further by among others, offering a graphical user interface around the import. The tasking manager contains essential information. Like a description of the import, instructions on how to do the import and which tags to use, etc. It provides an easy way of downloading a dataset, bounded by a grid, into JOSM or id editor.

The New York building import took ten months, finishing in June 2014. The project started as a community import but underestimating the import complexity and time spent training and supporting new mappers they restarted a few months in, loosely forming a group around the project [Barth, 2014b]. The group consisted of volunteer mappers and employees from the Mapbox team. This grouping made coordination easier and also made it simpler to ensure proper training [Barth, 2014a]. More than 20 people spent more than 1500 hours, importing 1 million buildings and over 900 000 addresses [Barth, 2014b]. Common issues during import were written on the Github page. The New York City data was first converted into OSM file format, then cut into byte sized blocks which were reviewed and imported manually through the tasking manager, piece by piece. An important validation step was that a different person than the original importer validated the data, reviewing it for errors and cleaning up when needed [Barth, 2014b].

The LA building import started in 2014. Two OSM enthusiasts started on the

project, Jon Schleuss, and Omar Ureta. They used code from the NY building import and adapted it to their needs. After a while, Mapbox joined in on the import, an important step for the project. Mapbox helped with important programming, creating scripts, converting the data to osm files, creating block groups of the data. The first challenge was to decide which datasets to import. They ended up neglecting address data, which would *delay the project with 1 year or 2* - quote Jon Schleuss, adding just building outlines and building info (assessor data) [Schleuss et al., 2016]. They merged and cleaned the datasets, splitting them into blocks and serving the data to the tasking manager. They used mapathons to get the import started. MaptimeLA was the organizer, and they also created tutorials for new mappers who joined the team. The first mapathons started with JOSM training to new mappers. Evolving to only arranging import mapathons, or import parties. Through mapathons is was easier for inexperienced mappers to contribute, here they could get the necessary training. When importing data mappers always have to examine for possible conflicts between existing and new data. If a conflict is detected, and the mapper doesn't know how to deal with it, they can flag the .osm file, and a more advanced user will look at it. The task will then be finished by someone else.

A big difference between the NY building import and LA building import was that the NY team ended up only allowing some OSM users to import. The NY-import was planned as a community import, but underestimating the import complexity and time spent training and supporting new mappers they restarted a few months in, they loosely formed a group around the project [Barth, 2014b]. The LA building team allowed everyone to join the import. To keep track of mappers the LA team created a list where the volunteers had to write their import username [**?**]. Doing the import job during mapathons is a good idea, making it easier to have an overall control over the import process.

In NY building import, when an error was detected that required updates to already imported data they had to do an automated edit. Updating existing data manually was very time-consuming. Updating OpenStreetMap data programmatically, with a script, is according to Alex Barth in Mapbox, crucial for a successful import. The LA community has pointed out errors that need to be fixed, for instance, is Garage incorrectly labeled as houses and condos have been tagged as a house, not as apartments. Errors encountered in already imported data are either fixed manually or through scripts, both cases are found reading through issues reported on the LA building GitHub page. The LA team created a Maproulette challenge on at least one issue, correcting split buildings. First, they implemented a script to detect all split buildings and then made each discovered building available as a task in Maproulette

[Sambale, 2016]. This approach was then using the mico-tasking method for solving errors.

When using the OSM tasking manager the dataset has to be divided into smaller parts. Each part represents one task, and it is important that each task is small enough so that it can be completed in a reasonable time, introduced in chapter 5. The NY team created a python script (chunk.py) to divide the data into smaller parts. The script divided the data into the New York City voting districts, there are in total 5258 voting districts, creating 5258 tasks in OSM Tasking Manager. Dividing the data into NY voting districts was an arbitrary choice, determined by the import team [Barth, 2014b]. The LA building mapper Alan McConchie opened an issue on LA-buildings GitHub page asking "*How to divide up the tasks?*" [McConchie, 2014]. He suggested using census block groups in each county, this grouping gave suitably sized areas for the tasks. Census blocks alone would be too granular tasks, and next level there are tracks, which would result in too big tasks. They used the same script as NY building import (chunk.py).

6.4 Evaluation

OpenStreetMap is a large community dependent on active users adding data in their geographic regions. The users have different perspectives on importing data through scripts. In empty regions with few active users, a bulk import can be a good way of developing the OSM project. But in regions with large amounts of data, there is no agreement on what's best*. Little research has been done in countries where the OSM project relied on data imports to fill the map. The OSM community is undecided on the benefits of bulk imports for the OSM project, especially for areas such as the US where large governmental data are freely available [Zielstra et al., 2013].

Validation of the data being imported takes time. Another dimension to bulk imports, with validation especially through tasking manager, is that unrelated issues in the same area get fixed by the mappers. During the New York building import, they fixed 5,000 unrelated map issues along the way [Barth, 2014b]. A very positive effect of the import. Another very positive effect of the NY building import was that from the beginning of the New York building import a goal was to help the city government maintain its building and address datasets [Barth, 2014c]. An edit in OSM can be a signal that the building has changed or the imported data is wrong. To help, the New York GIS department subscribes to daily email notifications to

building and address changes in OSM. A very good example of how government and open source *can take advantage of each other.

Does the OSM Tasking Manager tool make the import of data into OSM easier for the users? Of course it makes the coordination easier. But looking in the technical perspective maybe not. Or it just tells us that importing data to OSM following all the import guidelines will never be easy. The users that do the import need technical background and deep understanding of how OSM works.

During the LA import, the LA County released new data. This new data was used for rest of the import, but they did not update the already imported regions. How old was the NY building data?

7 | Suggested method for importing FKB into OpenStreetMap

Proof of concept

7.1 Micro-tasking method?

Chapter 6 introduce three common import methods, one of them is the micro-tasking method. The micro-tasking method stand out as the best alternative when considering the three methods. Especially together with a micro-tasking tool like the OSM Tasking Manager. The method and tool together solve problems other import methods face. When using them it is not necessary to create a personal dev-website with a list of an import queue like Dave Hansen did during the Tiger import **??** or creating a personal drive folder with the import files, like the OSM user tibnor did during N50 import. The micro-tasking method with the correct tool simplifies the import process.

The micro-tasking method is of course dependent on a good tool that organize the tasks well. The tasks should be easy to select, to download, to comment on and to mark as resolved. Information about the import and how to complete the tasks should be available and easy to understand.

The first challenge when using the micro-tasking method is dividing the dataset into smaller parts which give manageable sized tasks. Dividing the building datasets was a challenge in both the New York- and Los Angeles building import, introduced in section 6.3. Both solved this issue by dividing it into smaller parts using already defined subregions.*When considering the FKB building datasets, instead of dividing it into predefined regions, another alternative is to create subregions by counting

buildings. The density of buildings varies between different areas in Norway. The counting approach would ensure that every task has a manageable size. The number of buildings within each task could be from 20-30 units. This number must off course be tested with a test import before a final decision is made. What's important is that finishing a task should not take more than a few hours maximum, but at the same time should not be too small either. Too small tasks can result in too many tasks. *

A huge advantage in both New York and Los Angeles import was Mapbox. Having a company involved with professional staff who gets paid is challenging to replace. Luckily the tools created during both imports is open source, available at GitHub. Before implementing any new tool, the FKB building import team should look at already created tools. For instance, the LA-building team developed a plugin for JOSM called *Auto-tools*. This tool makes it much easier to combine two overlapping buildings.*

Another challenge that will emerge in a import is overlapping buldings. There are already buildings on the OSM-map of Norway, for instance shown in figure 1.1 and 3.1. These buildings are manually drawn and added by OSM users. Handling the conflicts according to the import guidelines is important. OpenStreetMap has no concept of layers, data on top of data will make it difficult for users to work in the standard OSM editors *.Overlapping layers needs to be merged. A reasonably safe assumption is that the building geometry is more accurate in FKB than in the manually drawn buildings. In JOSM there are a replace geometry tool. This tool replaces the geometry of the old building with the geometry of the new building while keeping all the tags and relations of the old one.*When the LA team worked on a task, they had at least two layers in JOSM, one with current OSM data and the other with the imported data. Both layers are reviewed for possible conflicts. If data in OSM was better than the imported data, only tags are transferred.

7.2 FKB to OSM tagging

An example of a area representation of a building feature type is shown in listing 7.1. This can be used as the building footprint when converting FKB to OSM. This will create the 2D modeling.

Listing 7.1: Example of a area representation of a building feature type in SOSI.

```
.FLATE 715235:
```

```
..OBJTYPE Bygning
..KOMM 1601
..BYGGNR 182720836
..BYGGTYP_NBR 111
..BYGGSTAT TB
..KOPIDATA
...OMRÅDEID 1601
...ORIGINALDATAVERT "Trondheim␣kommune"
...KOPIDATO 20160502
..REF :166806
..NØ
703610900 55898600
```

Using figure 7.1 of the twenty most common building types in Trondheim, there are in total about 140 different types [SOSI-sekretariatet,]. Not all building types used in FKB can directly translate to OSM. In the taginfo web page users can search for values commonly used on the building key. The taginfo page was helpful when transferring the FKB building types over to building values in OSM. For instance, type 181 is garage, 111 is detached house (*enebolig*) and 121 is house and are the three most common building types in Trondheim. A list of the 40 most common building types transferred into building values in OSM is in the appendix, table 9.1 and 9.1.

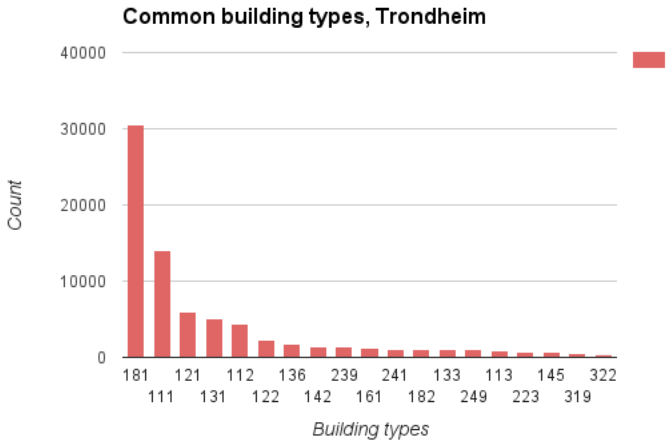

Figure 7.1: 20 most common building types in Trondheim, source: postgis query in the cadastre

These values will represent the value for the building key given to the building feature type. The building feature in listing 7.1 will be given the value detached (building=detached).

7.3 Conversion using existing script

The sosi2osm script was developed in 2013 by the github user Gnonthgol. It was created during the N50 imports initial stages.*The script is open source and the repository is available at Gnonthgol's GitHub account. The sosi2osm repository do not have any documentation, the only available help is a wiki page who shortly explain how to install and run the code. It is very difficult to install, especially if you don't have a linux operating system.

The script depends on fyba. An open source code distributed by the National Mapping Authority of Norway (Kartverket) to read and write SOSI files. Sosi2osm do not support SOSI files encoded in UTF-8. This is a challenge since FKB sosi files can be UTF-8 encoded.*This is a major drawback with this script. To test the script, after managing to install it correctly, the first step is to encode the FKB SOSI file into ISO8859-10. This step should not have been necessary.

When running the script, input is a sosi file and a lua script. A new lua script needs to be created for every conversion desired. In the N50 import they created a lua file for land cover (*arealdekke*), there is also a lua file for address import. The lua file creates key-value pairs from the metadata of the input file. Important metadata in sosi files is the feature type ($OBJTYPE$). In FKB building another important attribute is $BYGGTYP\_NBR$ which give information of what kind of building it is.

To test the sosi2osm script on the FKB building dataset a fkbbuilding.lua file was created. Using only the most common feature type in the dataset shown in figure 4.1*.Adding values to the building key, which depend on the buildingtype code from FKB, I used table 9.1 and 9.1. See listing 9.1 for the code snip checking the building value. This code snip checks the $BYGGTYP\_NBR$ attribute value to determine the buildings particular usage. The building=* key-value pair will be placed on the buildings footprint, since only building feature in FKB has this attribute value.

It is not possible to add height in ways and node representations. Another problem with the sosi2osm script is that it do not consider height values, creating one node for

each north, east coordinate pair. If two crossing building lines have different height values they should not share the same node [OpenStreetMap, g]. This is a problem, especially when considering 3D modeling of buildings.

7.4 How to map FKB buildings in 3D

In order to create a XML representation capable of modeling FKB buildings in 3D, a standard approach should be developed. Members of the OpenStreetMap community, with interest in 3D mapping, started in March 2012 to unite all the separated approaches to model 3D buildings using OSM XML [OpenStreetMap, 2013]. They arranged workshops, which resulted in a suggestion for a simple 3D building schema. This is the approach mentioned in section 3.5. This approach is fairly easy to implement, if the building's roof shape is known. This is not the case for the FKB buildings, so the simple 3D building schema needs modifications. Buildings in FKB is modelled with ridge and edge lines and they can be used to create 3D models, see the figures in section 4.3. When using ridge and edge modeling, roof shape tags are ignored, meaning the shape of the roof is not needed. Collecting ridge- and edge-lines for one building, creating a way-representation for each line with roof:edge and roof:ridge key's. The way-tag representing the building outline, most often the roof-edge feature, holds the height and roof:height information.*The listing shown in 9.2 creates a 3D representation of a house in OSM from FKB data. The house is shown in figure 7.2, using the JOSM editor to generate the code in listing 9.2 and getting 3D visualization using the JSOM plugin kendzi3d.

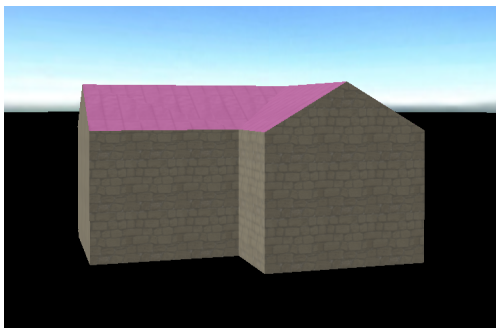

Figure 7.2: 3D representation of a FKB building, result of listing 9.2

In OpenStreetMap, building height is the distance between the lowest possible position

with ground contact and the top of the roof of the building, excluding antennas, etc [OpenStreetMap, 2016]. Height of buildings in FKB is height above sea level. This is a new challenge when mapping FKB buildings in OSM. In listing 9.2 height above sea level is manually withdrawn from the height given from the FKB data.

The problem with not distinguishing between overlapping nodes with different heights, mentioned in section 7.3, make 3D modeling of buildings with overlapping roof ridges and edges a challenge. This problem is shown in figure 7.3. Here it's easy to see the effect of overlapping points, they should be located in different heights, but share nodes when using the sosi2osm script.

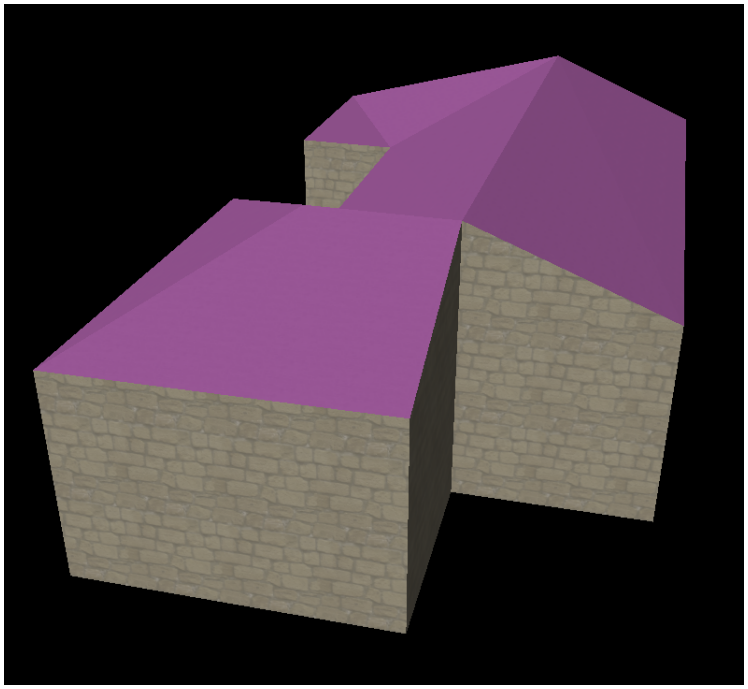

Figure 7.3: 3D representation of a FKB building

There are different tools developed to visualizing 3D buildings with data from OSM. A problem is that not all tools supports the different modelling. A list of which of them who supports the simple 3D building schema is located at the OSM simple 3D buildings wiki page, most tools accept this schema. 7 tools supports building:part=yes 2D renderers ignore building:part=* tags, only displaying the building outline.

7.5 Evaluation

From the beginning of the New York building import a goal was to help the city government maintain its building and address datasets [Barth, 2014c]. An edit in OSM can be a signal that the building has changed or the imported data is wrong. To help, the New York GIS department subscribes to daily email notifications to building and address changes in OSM. A very good example of how government and open source *can take advantage of each other.

Bibliography

[Barth, 2014a] Barth, A. (2014a). [Imports-us] Restarting NYC building and address imports. https://lists.openstreetmap.org/pipermail/imports-us/2014-February/000521.html.

[Barth, 2014b] Barth, A. (2014b). OpenStreetMap | lxbarth sin dagbok | Importing 1 million New York City buildings and addresses. http://www.openstreetmap.org/user/lxbarth/diary/23588.

[Barth, 2014c] Barth, A. (2014c). Over 1 million New York City buildings and addresses imported to OpenStreetMap | Mapbox. https://www.mapbox.com/blog/nyc-buildings-openstreetmap/.

[Bernstein et al., 2013] Bernstein, M., Bright, M., Cutrell, E., Dow, S., Gerber, E., Jain, A., and Kulkarni, A. (2013). Micro-volunteering: Helping the Helpers in Development.

[Bernstein et al., 2015] Bernstein, M. S., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., Crowell, D., and Panovich, K. (2015). Soylent: A Word Processor with a Crowd Inside. *COMMUNICATIONS OF THE ACM*, 58(85).

[Bishop, 2007] Bishop, D. (2007). [OSM-talk] TIGER, which states next? https://lists.openstreetmap.org/pipermail/talk/2007-October/019228.html.

[Chilton et al., 2009] Chilton, S., Building, F., and Burroughs, T. (2009). CROWDSOURCING IS RADICALLY CHANGING THE GEODATA LANDSCAPE : CASE STUDY OF OPENSTREETMAP.

[community OpenStreetMap,] community OpenStreetMap. Elements - OpenStreetMap Wiki. https://wiki.openstreetmap.org/wiki/Elements.

[Debruyne et al., 2015] Debruyne, C., Panetto, H., Meersman, R., Dillon, T., Weichhart, G., An, Y., and Ardagna Afostino, C. (2015). On the Move to Meaningful Internet Systems: OTM 2015 Conferences. *Springer International Publishing*.

[edits OpenStreetMap,] edits OpenStreetMap, A. Automated edits - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/Automated_edits.

[Exel van, 2010] Exel van, M. (2010). Data Imports In OpenStreetMap – Love 'Em Or Loathe 'Em? | oegeo. https://oegeo.wordpress.com/2010/10/22/data-imports-openstreetmap/.

[Exel van, 2013] Exel van, M. (2013). State of the Map US: San Francisco 2013 - MapRoulette, one year later. https://vimeo.com/68098505.

[Freyfogle, 2016] Freyfogle, E. (2016). OpenCage Data Blog — Open Geo interview - the makers of WeeklyOSM. http://blog.opencagedata.com/post/145261049883/open-geo-interview-the-makers-of-weeklyosm.

[Gnonthgol, 2013] Gnonthgol (2013). [NUUG kart] sosi2osm. https://lists.nuug.no/pipermail/kart/2013-October/004314.html.

[Gutwin et al., 2004] Gutwin, C., Penner, R., and Schneider, K. (2004). Group Awareness in Distributed Software Development. pages 72–81.

[Hagen, 2014] Hagen, S. O. (2014). Re: [NUUG kart] Møsvatn. https://www.mail-archive.com/kart@nuug.no/msg01508.html.

[Hansen, 2007a] Hansen, D. (2007a). [OSM-talk] TIGER update (need more suggestions). https://lists.openstreetmap.org/pipermail/talk/2007-November/019880.html.

[Hansen, 2007b] Hansen, D. (2007b). [OSM-talk] TIGER upload queue (maps of entire US). https://lists.openstreetmap.org/pipermail/talk/2007-September/017621.html.

[Hansen, 2007c] Hansen, D. (2007c). [OSM-talk] TIGER upload queue (maps of entire US). https://lists.openstreetmap.org/pipermail/talk/2007-September/017744.html.

[HOT Tasking Manager, 2016] HOT Tasking Manager (2016). HOT Tasking Manager - #2261 - Tanzania Development Trust: Serengeti district mapping project part 2. http://tasks.hotosm.org/project/2261.

[Jørgenrud, 2013] Jørgenrud, M. (2013). Kartskatt til alle - Digi.no. http://www.digi.no/artikler/kartskatt-til-alle/286539.

[Kartverket, a] Kartverket. Om Geovekst-samarbeidet | Kartverket. http://www.kartverket.no/geodataarbeid/geovekst/Om-Geovekst-samarbeidet/.

[Kartverket, b] Kartverket. SOSI | Kartverket. http://www.kartverket.no/geodataarbeid/standarder/sosi/.

[Kartverket, 2006] Kartverket (2006). SOSI standard -versjon 4.0 1 Del 1: Realisering i SOSI-format og GML SOSI Del 1: Realisering i SOSI-format og GML.

[Kartverket, 2011] Kartverket (2011). SOSI Del 3 Produktspesifikasjon for FKB – Generell del Side 2 av 55.

[Kartverket, 2013a] Kartverket (2013a). Samletabell over objekttyper i FKB.

[Kartverket, 2013b] Kartverket (2013b). SOSI Del 3 Produktspesifikasjon for FKB – Bygning.

[Kartverket, 2016a] Kartverket (2016a). Innføring i kommunene | Kartverket. http://www.kartverket.no/Prosjekter/Sentral-felles-kartdatabase/sentral-lagring-av-fkb-data–innfor

[Kartverket, 2016b] Kartverket (2016b). SOSI-standard del 2 Generell objektkatalog. http://www.kartverket.no/geodataarbeid/standarder/sosi/sosi-standarden-del-2/.

[Kartverket, c] Kartverket, S.-s. Bygningstype - Geonorge objektregister. https://objektkatalog.geonorge.no/Objekttype/Index/EAID_7C268089_ED2E_494b_9982_FD19

[Kartverket, d] Kartverket, S.-s. TreDNivå - Geonorge objektregister. https://objektkatalog.geonorge.no/Objekttype/Index/EAID_1DA8CD10_FAF4_4d0f_A46B_BB

[Kartverket, e] Kartverket, S.-s. Veranda - Geonorge objektregister. https://objektkatalog.geonorge.no/Objekttype/Index/EAID_3C31BCFA_6CE5_4af1_BB76_F7D

[Kihle, 2014] Kihle, K. (2014). [NUUG kart] Bruk av datasett fra Kartverket i OpenStreetMap. https://lists.nuug.no/pipermail/kart/2014-August/004831.html.

[Lidman, 2014] Lidman, A. (2014). ICCM and micro-tasking tools in OpenStreetMap | Mapbox. https://www.mapbox.com/blog/international-conference-of-crisis-mappers-2014/.

[Madalena and Clara, 2015] Madalena, M. and Clara, B. (2015). The Strengths and Weaknesses of Micro-volunteering: The Case Study of Help From Home.

[McAndrew, 2015] McAndrew, J. (2015). New York City | State of the Map US 2015. http://stateofthemap.us/2015/less-bots-more-humans-using-maproulette-to-import-data/.

[McConchie, 2014] McConchie, A. (2014). How to divide up the tasks? #4 Github issues, LA buildings. https://github.com/osmlab/labuildings/issues/4.

[Mehus, 2014] Mehus, T. (2014). [Imports] N50 imports from Kartverket (The Norwegian Mapping Authority). https://lists.openstreetmap.org/pipermail/imports/2014-June/003252.html.

[Mielczarek, 2007] Mielczarek, T. (2007). [OSM-talk] [OSM-dev] TIGER, which states next? https://lists.openstreetmap.org/pipermail/talk/2007-October/019231.html.

[Munro, 2007] Munro, R. J. (2007). [OSM-talk] TIGER upload status since 0.5 api. https://lists.openstreetmap.org/pipermail/talk/2007-October/018834.html.

[Neis and Zipf, 2012] Neis, P. and Zipf, A. (2012). Analyzing the Contributor Activity of a Volunteered Geographic Information Project — The Case of OpenStreetMap. *ISPRS International Journal of Geo-Information*, 1(3):146–165.

[ODbL,] ODbL. Open Database License (ODbL) v1.0 | Open Data Commons. http://opendatacommons.org/licenses/odbl/1.0/.

[OpenStreetMap, a] OpenStreetMap, C. Automated Edits code of conduct - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/Automated_Edits_code_of_conduct.

[OpenStreetMap, b] OpenStreetMap, C. building | Keys | OpenStreetMap Taginfo. http://taginfo.osm.org/keys/building#values.

[OpenStreetMap, c] OpenStreetMap, C. Changeset - OpenStreetMap Wiki. https://wiki.openstreetmap.org/wiki/Changeset.

[OpenStreetMap, d] OpenStreetMap, C. LearnOSM. http://learnosm.org/en/osm-data/file-formats/.

[OpenStreetMap, e] OpenStreetMap, C. MapRoulette - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/MapRoulette.

[OpenStreetMap, f] OpenStreetMap, C. Node - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/Node.

[OpenStreetMap, g] OpenStreetMap, C. Node - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/Node.

[OpenStreetMap, h] OpenStreetMap, C. Relation - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/Relation.

[OpenStreetMap, i] OpenStreetMap, C. State Of The Map - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/State_Of_The_Map.

[OpenStreetMap, j] OpenStreetMap, C. Way - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/Way.

[OpenStreetMap, 2007] OpenStreetMap, C. (2007). TIGER - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/TIGER.

[OpenStreetMap, 2013] OpenStreetMap, C. (2013). Simple 3D buildings - One year experiences / OpenStreetMap 3D / OpenStreetMap Forum. https://forum.openstreetmap.org/viewtopic.php?id=19729.

[OpenStreetMap, 2014] OpenStreetMap, C. (2014). Import/Catalogue/N50 import (Norway) - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/Import/Catalogue/N50_import_(Norway).

[OpenStreetMap, 2016] OpenStreetMap, C. (2016). Simple 3D buildings - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Simple_3D_buildings&oldid=1389188.

[OpenStreetMap, k] OpenStreetMap, D. Data working group - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/Data_working_group.

[OpenStreetMap, l] OpenStreetMap, I. Import - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/Import.

[OSM Tasking Manager,] OSM Tasking Manager, L. B. OSM Tasking Manager - LA buildings import. http://labuildingsimport.com/.

[OSMF,] OSMF. About | OpenStreetMap Blog. https://blog.openstreetmap.org/about/.

[OSMF, 2015] OSMF (2015). Membership/Statistics - OpenStreetMap Foundation Wiki. https://wiki.osmfoundation.org/wiki/Membership/Statistics.

[OSMstats, 2016] OSMstats (2016). OSMstats - Statistics of the free wiki world map. http://osmstats.neis-one.org/.

[Palen et al., 2015] Palen, L., Soden, R., Anderson, T. J., and Barrenechea, M. (2015). Success & Scale in a Data-Producing Organization. *Proceedings of the*

33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15, pages 4113–4122.

[Reiter and Barroso, 2016] Reiter, M. and Barroso, L. (2016). WeeklyOSM: Why, how, what, where and where we go. http://2016.stateofthemap.org/2016/weeklyosm-why-how-what-where-and-where-we-go/.

[Sambale, 2016] Sambale, M. (2016). OpenStreetMap | manings sin dagbok | Fixing split buildings in LA. http://www.openstreetmap.org/user/manings/diary/39230.

[Schleuss et al., 2016] Schleuss, J., Ureta, O., McConchie, A., and Sambale, M. (2016). Let's get LA on the map!: The Los Angeles Building Import Case Study | Seattle, Washington | State of the Map US 2016.

[Skogseth and Norberg, 2014] Skogseth, T. and Norberg, D. (2014). *Grunnleggende landmåling*. 1 edition.

[Soden and Palen, 2014] Soden, R. and Palen, L. (2014). From Crowdsourced Mapping to Community Mapping: The Post-earthquake Work of OpenStreetMap Haiti.

[Solhagen, 2015] Solhagen (2015). User:Solhagen - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/User:Solhagen.

[SOSI-sekretariatet,] SOSI-sekretariatet. BygningstypeKode - Geonorge objektregister. https://objektkatalog.geonorge.no/Objekttype/Index/EAID_A408BF17_3017_41c

[TagInfo, 2016] TagInfo (2016). building | Keys | OpenStreetMap Taginfo. http://taginfo.openstreetmap.org/keys/building.

[Wang et al., 2013] Wang, M., Li, Q., Hu, Q., and Zhou, M. (2013). Quality analysis of open street map data. *8th International Symposium on Spatial Data Quality*, 40(June):155–158.

[Willis, 2007] Willis, N. (2007). OpenStreetMap project imports US government maps | Linux.com | The source for Linux information. https://www.linux.com/news/openstreetmap-project-imports-us-government-maps.

[Willis, 2008] Willis, N. (2008). OpenStreetMap project completes import of United States TIGER data | Linux.com | The source for Linux information. https://www.linux.com/news/openstreetmap-project-completes-import-united-states-tiger-data.

[Zielstra et al., 2013] Zielstra, D., Hochmair, H. H., and Neis, P. (2013). Assessing the effect of data imports on the completeness of openstreetmap - A United States case study. *Transactions in GIS*, 17(3):315–334.

8 | Source - OSM mailing lists

The mailing lists and threads used as information source in this paper.

1. NUUG kart, mailing threads:
   - sosi2osm [Gnonthgol, 2013]
   - Bruk av datasett fra Kartverket i OpenStreetMap [Kihle, 2014]
   - Moesvatn [Hagen, 2014]
2. OSM-talk, mailing threads:
   - TIGER upload status since 0.5 api [Munro, 2007]
   - TIGER, which states next? [Mielczarek, 2007]
   - TIGER update (need more suggestions) [Hansen, 2007a]
   - TIGER upload queue (maps of entire US) [Hansen, 2007c]
3. Imports, mailing threads:
   - N50 imports from Kartverket [Mehus, 2014]

9 | Appendix

9.1 FKB building type to OSM building value

FKB: building type	count	OSM: building=*
181 - Garasje	30451.0	Garage
111 - Enebolig	13989.0	Detached
121 - Del av tomansbolig	6022.0	House
131 - Del av rekkehus	5143.0	House
112 - Enebolig	4351.0	Detached
122 - Tomannsbolig	2301.0	House
136 - Småhus	1705.0	House
142 - Boligbygg (3-4 etg)	1482.0	Appartment
239 - Lagerbygning	1380.0	Warehouse
161 - Fritidsbygg	1301.0	Cabin
241 - Hus for dyr	1111.0	Farm_auxiliary
182 - Garasje 1060.0	Garage	
133 - Atriumshus	1028.0	House
249 - Landbruksbygning	1007.0	Agricultural
113 - Våningshus	796.0	Farm
223 - Transformator	750.0	Substation
145 - Boligbygg (3-4 etg)	741.0	Appartment
319 - Kontorbygning	514.0	Office
322 - Butikk/forretningsbygg	365.0	Commercial

Table 9.1: Translating FKB buildingtypes to OpenStreetMap values, part 1

143 - Boligbygg (5 el. fler etg.)	310.0	Appartment
231 - Lagerhall	309.0	Warehouse
329 - Forretningsbygning	303.0	Commercial
199 - Annen boligb.	300.0	Yes
141 - Boligbygg (2 etg.)	290.0	Detached
612 - Barnehage	255.0	Kindergarten
183 - Naust, båthus, sjøbu	252.0	Boathouse
152 - Studentboliger	238.0	Dormitory
146 - Boligbygg (5 etg.)	218.0	Appartment
219 - Industribygning	213.0	Industrial
212 - Verkstedbygning	197.0	Service
619 - Skolebygning	173.0	School
144 - Boligbygg (2 etg.)	162.0	Apartment
243 - Veksthus	160.0	Greenhouse
613 - Barneskole	135.0	School
659 - Idrettsbygning	125.0	Sports_hall
123 - Del av våningshus, bolighus m/ to boliger	112.0	House
135 - Terrassehus	97.0	Apartment
211 - Fabrikkbygning	95.0	Factory
531 - Restaurantbygning	88.0	Restaurant
311 - Kontorbygning	83.0	Office

Table 9.2: Translating FKB buildingtypes to OpenStreetMap values, part 2

9.2 Code snip from .lua file, setting building=* value

Listing 9.1: Code snip from Lua file

```
elseif tokens[1] == "BYGGTYP_NBR" then
    if tokens[2] == "181" or tokens[2] == "182" then
        out["building"] = "garage"
    elseif tokens[2] == "111" or tokens[2] == "112" or tokens[2] == "141"
        out["building"] = "detached"
    elseif tokens[2] == "121" or tokens[2] == "123" or tokens[2] == "131"
        out["building"] = "house"
    elseif tokens[2] == "135" or tokens[2] == "142" or tokens[2] == "143"
        out["building"] = "appartment"
```

```
elseif tokens[2] == "239" or tokens[2] == "231" then
    out["building"] = "warehouse"
elseif tokens[2] == "161" then
    out["building"] = "cabin"
elseif tokens[2] == "241" then
    out["building"] = "farm_auxiliary"
elseif tokens[2] == "241" then
    out["building"] = "farm_auxiliary"
elseif tokens[2] == "249" then
    out["building"] = "agricultural"
elseif tokens[2] == "113" then
    out["building"] = "farm"
elseif tokens[2] == "223" then
    out["building"] = "substation"
elseif tokens[2] == "319" then
    out["building"] = "office"
elseif tokens[2] == "322" or tokens[2] == "329" then
    out["building"] = "commercial"
elseif tokens[2] == "612" then
    out["building"] = "kindegarten"
elseif tokens[2] == "183" then
    out["building"] = "boathouse"
elseif tokens[2] == "152" then
    out["building"] = "dormitory"
elseif tokens[2] == "219" then
    out["building"] = "industrial"
elseif tokens[2] == "212" then
    out["building"] = "service"
elseif tokens[2] == "613" or tokens[2] == "619" then
    out["building"] = "school"
elseif tokens[2] == "243" then
    out["building"] = "greenhouse"
elseif tokens[2] == "659" then
    out["building"] = "sports_hall"
elseif tokens[2] == "211" then
    out["building"] = "factory"
elseif tokens[2] == "531" then
    out["building"] = "restaurant"
```

```
elseif tokens[2] == "311" then
    out["building"] = "office"
else
    ut["building"] = "yes"
end
```

9.3 XML code: 3D representation of a house from FKB data

Listing 9.2: 3D representation of a house from FKB data in OSM XML

```
<node id="-161645" lat="63.4485538" lon="10.1827768" version="1" visible="true"/>
<node id="-387759" lat="63.4485656" lon="10.1827249" version="1" visible="true"/>
<node id="-387760" lat="63.4484758" lon="10.1826237" version="1" visible="true"/>
<node id="-387761" lat="63.4484641" lon="10.1826756" version="1" visible="true"/>
<node id="-387762" lat="63.4484517" lon="10.1827306" version="1" visible="true"/>
<node id="-387763" lat="63.4484944" lon="10.1827787" version="1" visible="true"/>
<node id="-387764" lat="63.4484820" lon="10.1828337" version="1" visible="true"/>
<node id="-387765" lat="63.4485052" lon="10.1828598" version="1" visible="true"/>
<node id="-387766" lat="63.4485289" lon="10.1828866" version="1" visible="true"/>
<node id="-387767" lat="63.4485411" lon="10.1828332" version="1" visible="true"/>

<way id="-166806" version="1" visible="true">
        <tag k="OMRÅDEID" v="1601"/>
        <tag k="KVALITET" v="24; 19; 0; 24; 23"/>
        <tag k="TRE_D_NIVÅ" v="2"/>
        <tag k="KURVE" v="166806"/>
        <tag k="OBJTYPE" v="Takkant"/>
        <tag k="building" v="yes" />
        <tag k="height" v="6.28" />
        <tag k="roof:height"  v="1.67"/>
        <tag k="building:colour" v="#a8a08e"/>
        <tag k="building:material" v="stone"/>
        <tag k="roof:colour" v="#d162ac"/>
        <tag k="roof:material" v="metal"/>
        <nd ref="-161645" />
        <nd ref="-387759" />
        <nd ref="-387760" />
        <nd ref="-387761" />
        <nd ref="-387762" />
```

```
        <nd ref="-387763" />
        <nd ref="-387764" />
        <nd ref="-387765" />
        <nd ref="-387766" />
        <nd ref="-387767" />
        <nd ref="-161645" />
</way>

<node id="-161645" lat="63.4485538" lon="10.1827768" version="1" visible="true"/>
<node id="-59094" lat="63.4485300" lon="10.1827499" version="1" visible="true"/>
<way id="-59454" version="1" visible="true">
        <tag k="OMRÅDEID" v="1601"/>
        <tag k="KVALITET" v="24; 19; 0; 24; 23"/>
        <tag k="TRE_D_NIVÅ" v="2"/>
        <tag k="KURVE" v="59454"/>
        <tag k="OBJTYPE" v="Mønelinje"/>
        <tag k="roof:ridge" v="yes" />
        <nd ref="-161645" />
        <nd ref="-59094" />
</way>
<node id="-161646" lat="63.4484641" lon="10.1826756" version="1" visible="true"/>
<way id="-59455" version="1" visible="true">
        <tag k="OMRÅDEID" v="1601"/>
        <tag k="KURVE" v="59455"/>
        <tag k="OBJTYPE" v="Mønelinje"/>
        <tag k="roof:ridge" v="yes" />
        <nd ref="-59094" />
        <nd ref="-161646" />
</way>
<node id="-161633" lat="63.4485052" lon="10.1828598" version="1" visible="true"/>
<way id="-59447" version="1" visible="true">
        <tag k="OMRÅDEID" v="1601"/>
        <tag k="ORIGINALDATAVERT" v="Trondheim kommune"/>
        <tag k="KURVE" v="59447"/>
        <tag k="OBJTYPE" v="Mønelinje"/>
        <tag k="roof:ridge" v="yes" />
        <nd ref="-59094" />
        <nd ref="-161633" />
</way>

<node id="-59093" lat="63.4485411" lon="10.1828332" version="1" visible="true"/>
```

```
<node id="-59094" lat="63.4485300" lon="10.1827499" version="1" visible="true"/>
<node id="-59095" lat="63.4484944" lon="10.1827787" version="1" visible="true"/>
<way id="-21118" version="1" visible="true">
        <tag k="OMRÅDEID" v="1601"/>
        <tag k="KURVE" v="21118"/>
        <tag k="OBJTYPE" v="Bygningslinje"/>
        <tag k="roof:edge" v="yes" />
        <nd ref="-59093" />
        <nd ref="-59094" />
        <nd ref="-59095" />
</way>
```