Evaluation of the Micro-Tasking Method for Importing High-Detail Building Models to OpenStreetMap

Anne Sofie Strand Erichsen

January 11, 2017

Contents

List of Figures

Abstract

Importing large quantities of data into the OpenStreetMap (OSM) database has always been challenging. This study will investigate if the micro-tasking method makes the import process more manageable for the OSM community. In order to evaluate this method, the detailed FKB building dataset covering Norway will be used as an example dataset. This dataset will, most likely, be freely available within few years. By using the FKB building dataset this study will also provide help to the community if or when an import of the FKB data is approved. This study also looks at how the buildings can be 3D modeled in OSM. The results show that the micro-tasking method should be the preferred method when doing large imports of external data into OSM.

1 | Introduction

OpenStreetMap (OSM) is an international effort to create a free source of map data through volunteer effort [Goodchild, 2007] and is the largest and probably the best-known example of geospatial data creation using Volunteered Geographic Information(VGI). VGI is described by Goodchild as a phenomenon where volunteer citizens manage to create and share geographic information through widespread access to the internet, handheld GPS devices, and computers. 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. The number of active users in different places 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 into the OSM database [Zielstra et al., 2013]. To get external data into the OSM database various import methods have been developed throughout the years, some more successful than others. No method seem to stand out as the preferred method in the OSM community. One method used today to import external data into OSM is the micro-tasking method. This study will examine the micro-tasking method and evaluate its strengths and weaknesses. At least two large-scale imports have used the micro-tasking method and they will be an important part of the evaluation in this paper.

In order to evaluate the micro-tasking method, this paper will use the FKB building datasets as test data. The Norwegian government has indicated that FKB, the most detailed map data of Norway, may become freely available within a few years. If so, this data may be imported into OpenStreetMap. FKB is a collection of datasets containing detailed vector data of Norway, created in the SOSI format. By using the FKB building data, this paper can also provide help to the OSM community, if or when an import of FKB into OSM is approved.

Importing data into OSM can be complicated and finding an 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 can be the core for a successful import method. The micro-tasking method could be the solution. Examining this method and how it could be used when importing the FKB building data is the core motivation for this study.

An import of FKB building data could result in a 3D model of buildings throughout Norway, free and available for everyone. FKB contain very detailed building data and gives a good foundation for 3D modeling of buildings. Trondheim will be a test municipality in this paper. The buildings on the OSM-map of Trondheim is today added manually by users drawing over aerial photos and by using their local knowledge. Height and roof shape is often not included; it's hard to get this information from an aerial photo. This makes the 3D models of buildings nonrealistic in OSM. An example of 3D models of buildings from OpenStreetMap today (late 2016) is shown in figure 1.1a. The same buildings 3D modeled with FKB building data is shown in figure 1.1b. Norkart created the FKB 3D model in figure 1.1b.

(a) From OpenStreetMap

(b) From FKB building data

Figure 1.1: 3D modeling of buildings in Trondheim, source: [Buildings, 2016a] and [NORKART, 2016]

In figure 1.1a the buildings have equal heights and a flat roof. This is not an accurate representation of the buildings in real life. Looking at figure 1.1b the buildings are modeled much more accurate. Finding out how to use FKB building data to create a realistic 3D modeling of buildings in OpenStreetMap will be the second motivation for this study.

2 | Background

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 [Weber, 2008]. OpenStreetMap was founded in 2004 at University College London by Steve Coast. The goal was to gather and share geospatial data of the whole world through a free geospatial database [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].

The OpenStreetMap Project is supported by the OpenStreeMap Foundation (OSMF) which is a UK-registered non-profit organization. OSMF is critical to the ongoing function and growth of the OpenStreetMap project. 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. [OSMF,]. 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,].

Communication between OSM users 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, 2016l]. Important communication is also done through both issues and pull requests in the repositories at the OpenStreetMap GitHub channel, which is easy to see in on the different OSM repositories.

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].

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 et al. 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. Creating a tasking manager 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]. This tool was the first introduction of the micro-tasking method in OpenStreetMap.

3 | Characteristics of OpenStreetMap

Important information when looking at how to import external data into OpenStreetMap is understanding the data structure of OpenStreetMap. This data structure is unique and only supported by OSM. This chapter will give an introduction to this structure. It will also cover how this data structure is realized through the XML format. When importing data into OSM the original data needs to be converted into XML files supporting this data structure. Since this study will look at FKB building data it is important to understand how to map buildings in OpenStreetMap, both in 2D and 3D. This will also be covered in this chapter.

3.1 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 [OpenStreetMap, 2016f]. 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 pages. On the *tags you like* wiki page they recommend different pages, 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 information on how frequent they appear in the OSM database. 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, 2016d]. A 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 edit in the OSM database is a part of a changeset.

3.2 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,].

Nodes, ways and relations is represented by their own tag: <node>, <way> and <relation>. 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, 2015a]. In listing 3.1 the node tag describes a bag shop named Citybag, this is called a point of interest. The key named user give the name of who last modified this node (user="Peter Bremer") and uid is 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. 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, 2016m]. 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 the building outline of a church since the key equals building and the value equals 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>
```

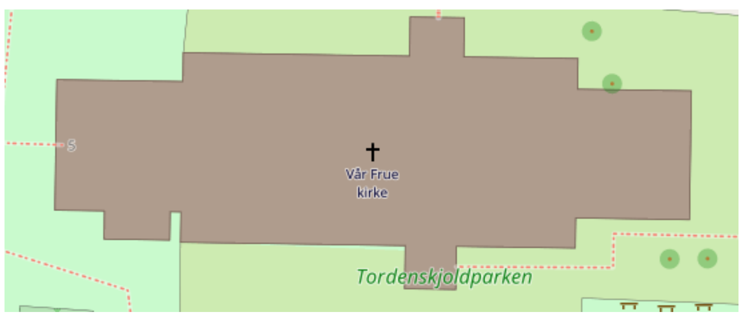


Figure 3.1: 2D representation of *Vaar frues kirke*, result of listing 3.2. Source: [OpenStreetMap, 2016a]

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, 2016j]. 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 the role as east, going towards the 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, used as 2D representation of the church. Listing 3.2 is the XML-code for this outline (notice that the ref number *89340594* 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 church shown in figure 3.2.

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

```xml
<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>
```

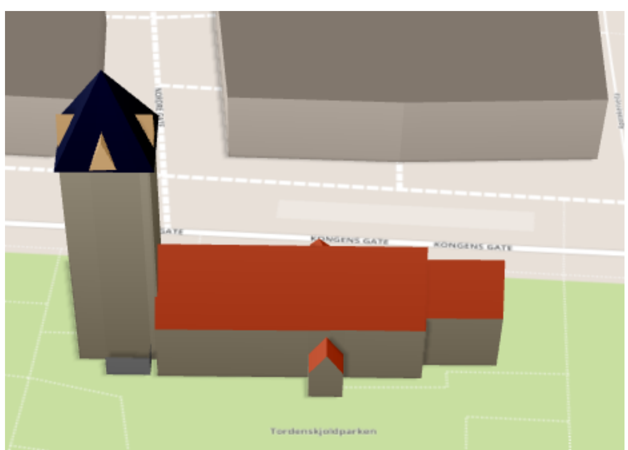


Figure 3.2: 3D representation of *Vaar frues kirke*, result of listing 3.3. Source: [Buildings, 2016b]

3.3 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 footprint of the building. 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, 2016c]. 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.

Relations is 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>
```

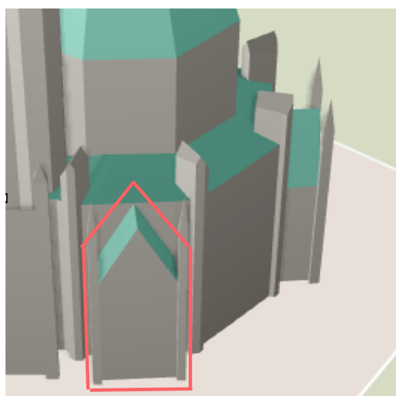


Figure 3.3: *Sydkapell* with gabled shaped roof, result of listing 3.4. Source: [Buildings, 2016b]

4 | SOSI and FKB building

This paper looks at the FKB building dataset and how it can be imported into OpenStreetMap. The FKB dataset comes in the 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]. 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, which was introduces in section 3.2.

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.

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 of 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 a 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 an FKB building dataset covering Trondheim municipality gives some indication of the most common feature types in Norway 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 delimitation (*bygningsavgrensning*), descriptive building lines, building appendage (*bygningsvedheng*) and lastly roof covering (*takoverbygg*). Some of the most common feature types, shown in figure 4.1, will be introduced to create a fundamental understanding of the FKB building data.

When considering an import into OpenStreetMap there will be features that are not as relevant. Adding all buildings as point data does not seem relevant. The building feature comes as both point- and area data types, containing exactly the same attributes. If the building footprint is available, points should not be imported, as mentioned in section 3.3. Therefore, the building feature represented as point data should not be imported into OSM. The building representation as area data should instead be used, creating the building footprint in OSM.

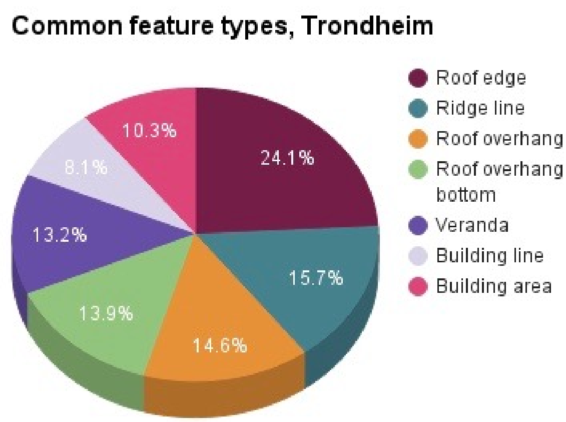


Figure 4.1: Most common feature types in Trondheim

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 delimitation (*avgrensning*). See figure 4.2 and 4.3 for examples on where this feature is mapped on buildings. If this feature is present it is used to form the building delimitation represented by the building feature. 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.

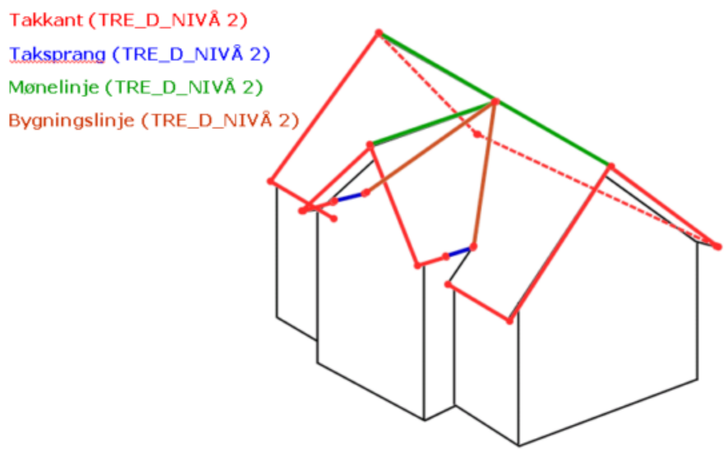


Figure 4.2: Roof edge (red), roof overhang (blue), ridge line (green) and building line (brown) [Kartverket, 2013b]

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.

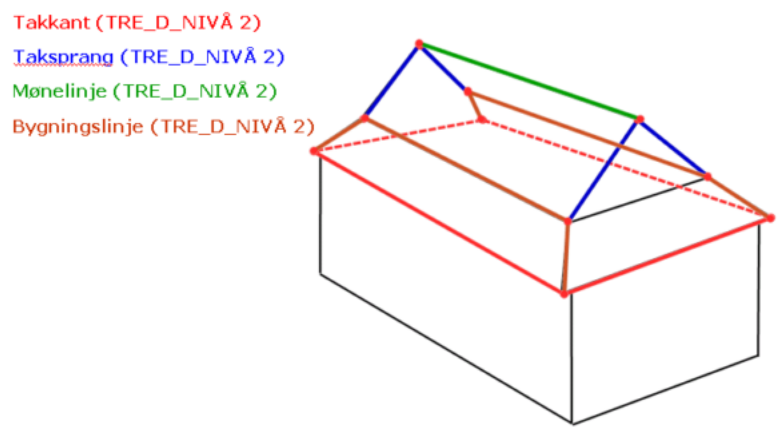


Figure 4.3: Roof edge (red), roof overhang (blue), ridge line (green) and building line (brown) [Kartverket, 2013b]

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 afterward to visualize roof overhang in the same figure.

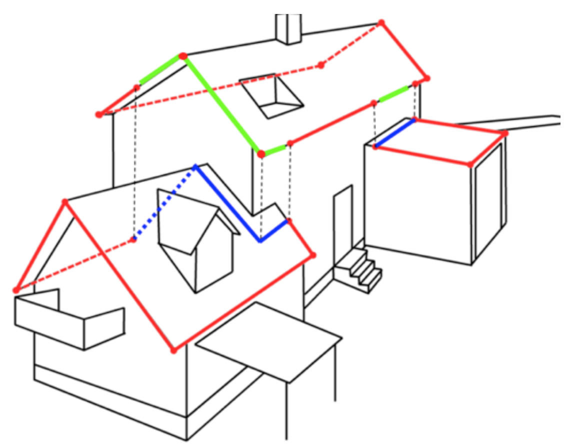


Figure 4.4: Roof overhang bottom (blue), roof edge (red) and roof overhang (green)

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.

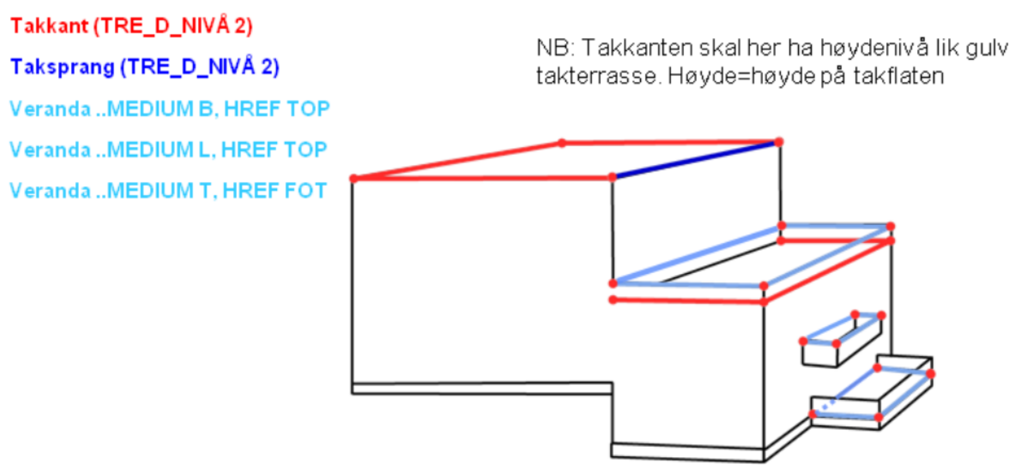


Figure 4.5: Roof edge (red), roof overhang (blue) and veranda (turquoise) [Kartverket, 2013b]

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. There are three types of building delimitation (*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 building delimitation. Roof outline inherit roof edge [SOSI-sekretariatet, b], making roof edge the foundation of the delimitation as mentioned earlier in this section. This feature also classifies the building through the attribute $BYGGTYP\_NBR$. This attribute contains an initial value which is mapped to a specific building type. House (*enebolig*) has initial value 111 and dormitory (*studenthjem*) value 152 [Kartverket, c] etc. More about this in section 4.4 and 7.2.

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

The SOSI format does not have polygons directly in its structure, a polygon must be represented with points and lines, and then connected through an area feature. This is quite similar to how the OSM XML format does it. The key-value pair structure used in OSM also has some similarities on how SOSI presents it's metadata, even though SOSI follows a standard way of representing the metadata, which OSM doesn't have.

Listing 4.1 is SOSI code for area representation of a building. $FLATE$ means area and it contains the feature id. OBJTYPE is feature type and in listing 4.1 equals building, representing what the key would be in OSM. The $KOMM$ 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 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, a]. 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 4.3. 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 of each point. KP1 means that this point is a connection point. 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 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 and is a screen shot from an aerial photo. 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 beneficial when modeling building parts in OSM, especially in 3D modeling, more about this in section 7.4. 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 the best possible import method is desired. A promising method when importing external data into OSM is micro-tasking. This chapter will give an introduction to the micro-tasking method. It will describe what micro-tasking is and introduce related tools using this method in OSM.

5.1 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].

5.2 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, 2016b].

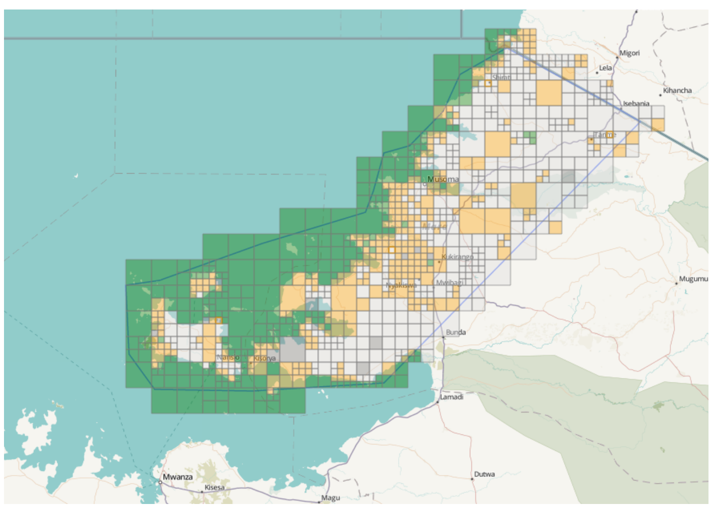


Figure 5.1: Project 2261- Tanzania Development, source: [HOT Tasking Manager, 2016a]

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. One new feature was adding possibilities for tasks being polygons and not grids, as shown in figure 5.2. They divided the tasks into reasonable size census block groups. More about this in subsection 6.3.

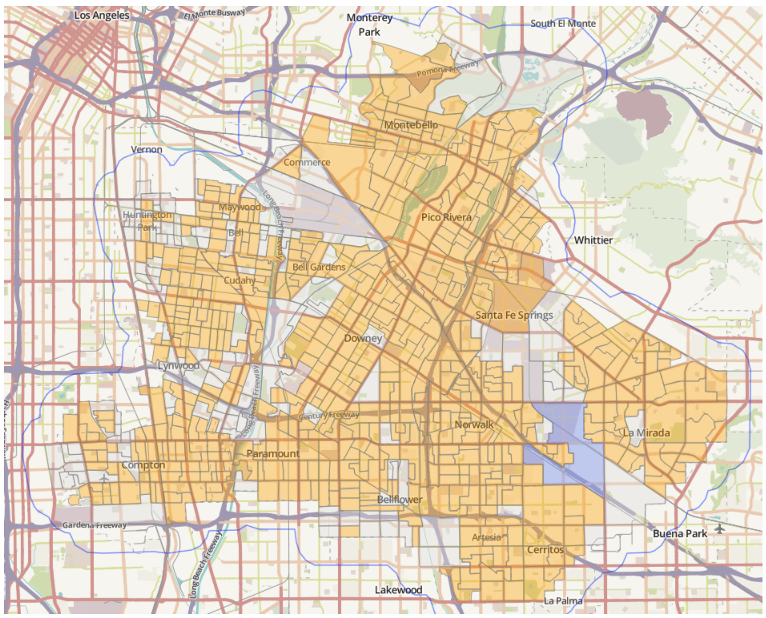


Figure 5.2: LABuildings Import, source: [Los Angeles building, 2016]

5.3 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, 2016i]. To-fix is a micro-tasking tool created by the company 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. Both tools convey statistics presenting total fixed and skipped taska, and also total tasks marked as not error. Both tools are good examples of how the micro-tasking method is beneficial for the OSM community.

6 | OSM import methods

To be able to evaluate methods of import and to conclude on if micro-tasking could be the desired method for bulk imports in OpenStreetMap, this study should examine other standard import methods as well. Therefore will this chapter evaluate three common import methods. A fully automatic method, a guided automatic method and the micro-tasking method. This chapter will try to give a good implication if the micro-tasking method can be the desired import method for the OSM community when importing external data.

6.1 Fully automatic import method

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 imports. 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, 2016]. This kind of edits must follow the Automated Edits code of conduct [OpenStreetMap, 2016b]. 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 limited oversight. "*Please don't upload Northampton County, [..] I've mapped my entire town there [..]*" [Mielczarek, 2007]. Active OSM mappers following the mailing list 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, 2007a]. He added states and counties after requests from users [Hansen, 2007b]. The import team did not have any validation or correction methods or routines. In the aftermath of the import, tagging errors were discovered. 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 TIGER 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 done through a fully automatic method are overall not recommended today. The lack of a validation process and the overwriting of existing data is a huge problem. A fully automatic import will for instance work on a empty map. Then no data is overwritten and there is no need for validation. 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 information.

6.2 Guided automatic import method

A fully automatic import of huge amounts of data is discouraged in the OSM community, so another method 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 it 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 the SOSI format and OSM file format, which he informed about in the sosi2osm mail thread [Gnonthgol, 2013]. This script is the recommended way of converting the N50 SOSI files since it converts directly from SOSI to OSM XML format. Benefits using the SOSI file format in the context of the XML format in OSM is mentioned in section 4.4.

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 existed in OSM, that conflicted with the new data was marked with a FIXME=Merge tag. Here the user had to search for the conflicting elements and correct the errors manually. Only after fixing the conflicting elements the data could be uploaded into 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, 2016e]. 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 bad 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 Micro-tasking import method

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 created by the HOT team.

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. The micro-tasking method together with the 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 a company named Mapbox. 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 datasets were first converted into the OSM file format, then cut into byte sized blocks which were reviewed and imported manually into OSM through the tasking manager, task by task. 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, the company Mapbox joined in on this import as well, an important step for the project. Mapbox helped with important programming, creating scripts, converting the data to osm files, and dividing the data into reasonable sized tasks [Schleuss et al., 2016]. 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 - 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, coordinated mapping events, to get the import started. The first mapathons started with training new mappers and evolving to only arranging import mapathons. Doing the import job during mapathons was a good idea, making it easier to have an overall control over the import process. 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 the volunteer mappers the LA team created a list on GitHub where the volunteers had to write their import username [Sambale, 2016]. Here there are about 60 registered volunteers today (December 2016).

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).

7 | Proposed method

Looking at an import of FKB building data into the OSM database, how should the process be done? What are the challenges? By using information from the previous chapters, this chapter will examine how to use the micro-tasking method to import FKB into OSM. To help the OSM community this chapter will also provide the reader with examples of how the FKB metadata could be mapped to OSM key-value pairs. It will also evaluate an existing conversion script and suggest improvements, which could be to help if a new conversion script is developed. At last, it will assess how to import FKB buildings data to achieve a 3D representation of the buildings in OSM.

7.1 Micro-tasking method?

Chapter 6 introduce three common import methods, one of them is the micro-tasking method. The micro-tasking method stands 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 for the OSM users participating in the import.

What is the correct tool? The micro-tasking method is, of course, dependent on a good tool that organizes 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 at all time and also be easy to understand. After some testing of the OSM Tasking Manager through this study, a safe assumption is that this tool solves the important parts mentioned above. Instead of having to read multiple import wiki pages, imports often have more than one information page, then follow email conversations about the import, ask which files to import next and after that having to find where the import files are located. The OSM Tasking Manager gathers all these steps in one place, making it much easier for users to contribute on imports.

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 20-40 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 will result in too many tasks.

Another challenge that will emerge in an import is overlapping buildings. There are already buildings on the OSM-map of Norway, for instance, shown in figure 1.1a 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 [OpenStreetMap, 2016h]. 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. This approach is a quite safe method to use on conflicting buildings. It is not too time-consuming and existing metadata is kept. A more time-consuming approach can be seen in the LA building import. 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 were reviewed for possible conflicts. If a conflict was detected the user examined if the data in OSM was better than the imported data. If the existing data was better only tags were transferred from the new data. This process is a more time-consuming approach but is probably a good solution when doing FKB building import in municipalities with a good coverage of buildings already. One should always consider the data already in OSM before importing new data.

A huge advantage in both New York and Los Angeles building import was the company 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 OSM community in Norway 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.

The micro-tasking method can attract more users to contribute in the import because it provides tasks with less complexity. An import of existing data into OpenStreetMap is not easy, as can be seen from previous imports. The N50 import team are still not finished, even though they started in 2015. It can be argued that their import method is too time-consuming. Dividing the data into .15 degree .15 degree changesets resulted in too big tasks, making each import file very time-consuming to import. Too large tasks will also make each import easily very complex, creating many conflicts to be fixed at the same time. Creating smaller tasks makes the import less complex because it's fewer conflicts to solve. This again can lower the boundaries for less experienced users to participate in the import. Looking at the user assignment list for the N50 import, about 19 OSM users are involved in the import. In comparison, the LA building import has about 60 users participating in the import [Building, 2016]. The OSM community in Los Angeles is probably larger than the community in Norway, but the number also hopefully reflect how the micro-tasking method reaches out to more users within the OSM community. Getting more volunteered people involved is important, especially if the import is not supported by a company who pay their employees to help like Mapbox did in both NY and LA building import. The import will take years if no company or if only experienced users can participate, as seen in the N50 import. By using smaller tasks the import also requires less time from the volunteers. If each task only requires, for instance, one hour, it is easier for users to find time to help the community finish the import.

Another positive effect of using the micro-tasking method is the OSM Tasking Manager tool. This tool provides all necessary information about the import on one web page. This makes it easier to spread the information about the import, instead of having to spread the information page, the page where the import files are located and also the page where the import queue is written. Having all necessary information and data gathered on one page makes it easier for users to get involved, especially for users who are not involved in the process from the beginning. This can also make it easier to get more people to join the import.

7.2 FKB metadata mapped to OSM key-value pairs

If FKB building data should get approval to be imported into OSM, the metadata in FKB must be mapped over to OSM key-value pairs. As introduced in section 3.1, the norm is to always try to use existing key-value pairs. Therefore will this section only uses key-value pairs that already exists in the proposed mapping.

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

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
```

Starting with building type, which is found under the attribute $BYGGTYP\_NBR$ in SOSI area representations. In listing 7.1 $BYGGTYP\_NBR$ equals 111, which should be mapped to building=detached in OSM. There are in total about 140 different building types in FKB [SOSI-sekretariatet, a]. Not all building types used in FKB can directly translate to OSM. In the taginfo web page (*taginfo.openstreetmap.org*) 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, see figure 7.5. A list of the 40 most common building types mapped to key-value pairs in OSM is listed in the appendix, see table 10.1 and 10.2. The conversions shown in table 10.1 and 10.2 are only this studies suggestions, the OSM community, especially the Norwegian, must approve the suggestions. The study only converted the 40 most common building types, to give the reader an impression of how it can be done.

The values in table 10.1 and 10.2 will represent the value for the building key. This information will, as mentioned, be added to the building outline. For the different feature types introduced in section 4.3 there are different approaches on how they should be mapped over to OSM. Using section 4.3 and figure 7.2 in section 7.4 the main feature types can be given the key-value pairs shown in table 7.1. More about this in section 7.4.

FKB: feature type	Key	Value
Roof edge (*Takkant*)	building	yes
Ridge line (*Mønelinje*)	roof:ridge	yes
Building line (*bygningslinje*)	roof:edge	yes
Roof overhang (*Taksprang*)	roof:edge	yes
Roof overhang bottom (*TaksprangBunn*)	roof:edge	yes

Table 7.1: Translating FKB feature types to OpenStreetMap key-value pairs

7.3 Conversion using existing script

Before implementing a new script to convert the FKB SOSI files to OSM XML-files, existing scripts should be evaluated. There is at least one script that converts SOSI files to OSM XML-files. This script was used by the N50 import team. The script is called sosi2osm and was developed in 2013 by the GitHub user Gnonthgol. The script is open source and the repository is available at Gnonthgol's GitHub account. The sosi2osm repository does 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.

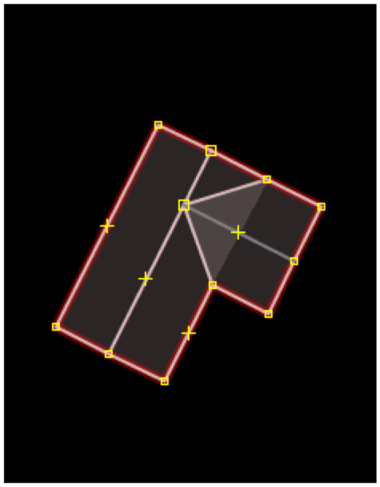
The script depends on fyba. An open source code distributed by the National Mapping Authority of Norway (Kartverket) to read and write SOSI files. Sosi2osm do not support SOSI files encoded in UTF-8. This is a challenge since FKB SOSI files can be UTF-8 encoded. This is a major drawback with this script. 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 the input is a SOSI file and a Lua script. A new Lua script needs to be created for every new dataset. The Lua script needs to fit the metadata present in the dataset the user is converting. In the N50 import, they created a Lua script for land cover (*arealdekke*) without water and one Lua script for only water [OpenStreetMap, 2016g]. The Lua script creates key-value pairs from the metadata of the input file. Important metadata in FKB is the feature type (*OBJTYPE*), introduced in section 4.3, and the $BYGGTYP\_NBR$ which give information of what kind of building it is, introduced in the previous section.

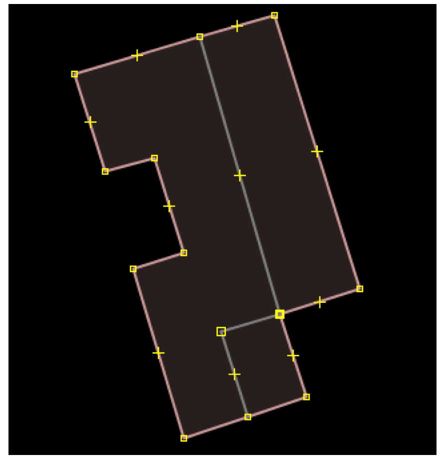
To test the sosi2osm script on the FKB building dataset a fkbbuilding Lua file was created. Using only the most common feature type in the dataset shown in figure 4.1. Adding values to the building key, which depend on the building type code in FKB, table 10.1 and 10.2 was used. See listing 10.1 in the appendix for a code snippet from the Lua script checking the building value. This code snippet checks the $BYGGTYP\_NBR$ attribute value to determine the buildings particular usage. The building=* key-value pair will be placed on the footprint of the building, since only building feature in FKB has the $BYGGTYP\_NBR$ attribute value.

A problem with the sosi2osm script is that it is not possible to retrieve height data from the SOSI file. It is then not possible to use this script to create 3D models. It does 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, 2015b]. This is also a problem, especially when considering 3D modeling of buildings. More about this in section 7.4.

The sosi2osm script managed to convert the FKB building dataset into a working OSM XML file using the fkbbuilding lua file. A code snippet from this Lua file is shown in listing 10.1 in the appendix, as mentioned above. The output file was tested using the JOSM editor. It worked for 2D representation of the buildings, in other words, created a functional representation of building outlines with the roof lines in it. Two sample buildings are shown in figure 7.1. The key-value pairs were set correctly and a relation tag was created for the building outline.



(a) Building footprint 1

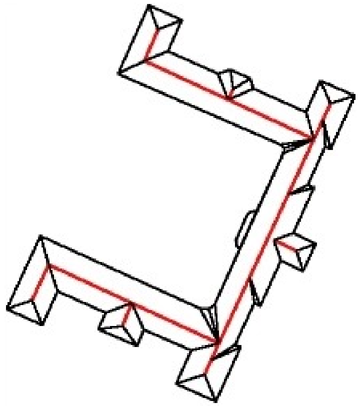


(b) Building footprint 2

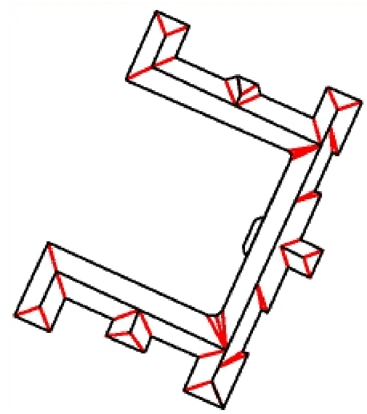
Figure 7.1: Building footprint with roof edges and roof ridges created with sosi2osm script

7.4 How to map FKB buildings in 3D

In order to create an XML representation capable of modeling FKB buildings in 3D, a standard approach, for every building type, is desired. Members of the OpenStreetMap community, with interest in 3D mapping, started in March 2012 to unite all the separate 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 3.3. 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.



(a) Roof ridges



(b) Roof edges

Figure 7.2: Roof lines in OpenStreetMap, source: [OpenStreetMap, 2015c]

Buildings in FKB is modeled 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 [OpenStreetMap, 2015d], meaning the shape of the roof is not needed. Using figure 7.2 as guidance. Ridge- and edge-lines for one building was collected to test this method and manually created a way-representation for each line with roof:edge and roof:ridge keys. The key-value pairs for each roof feature line are shown in table 7.1. The way-tag representing the building outline, which is most often the roof-edge (*takkant*) feature as mentioned in section 4.3, must hold the building=yes, height, and roof:height information. This of course only works on buildings where the whole building outline is only covered by the roof edge, this is not always the case.

Listing 10.2 in the appendix creates a 3D representation of a house in OSM from FKB data using the method described above. The house is shown in figure 7.3, using the JOSM editor to generate the code in listing 10.2 and getting 3D visualization using the JSOM plugin kendzi3d. The building outline in this building is only covered by the roof edge, meaning it's SOSI area representation only refers to one roof edge. The area representation of the building in figure 7.3 in SOSI format is shown in listing 7.1, notice that it only refers to one feature type (REF :166806). The feature with id=166806 is a roof edge. Then the technique described above works.

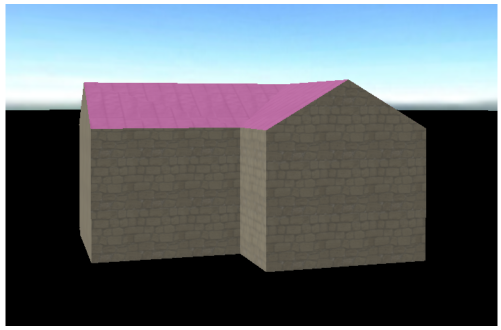
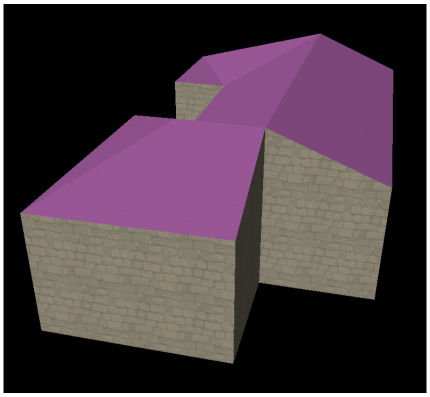


Figure 7.3: 3D representation of a FKB building, result of listing 10.2

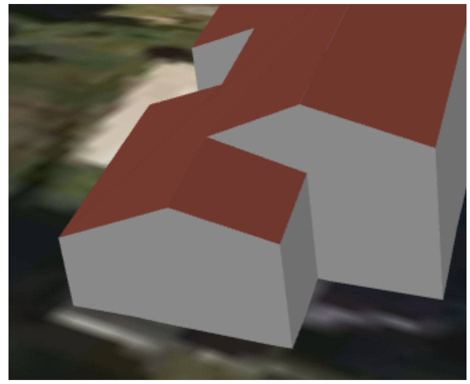
Getting the method working for buildings referring to more than one feature is more challenging, especially when the goal is to create a script that creates a working 3D model for every building type. A solution could be to gather all the features the building refers to in the SOSI file and join then in one way representation in OSM XML. This was tested manually, and it worked for the test objects. It's, of course, difficult to test every different case. Finding an approach that can manage every building type will take time.

Getting the correct height of the buildings is also a challenge. 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, 2016k]. The height of buildings in FKB is height above sea level. In listing 10.2 height above sea level is manually withdrawn from the height given from the FKB data. A solution is to use a digital elevation model to examine the elevation at every building. This will of course again cause challenges, for instance, if a house is built into a hillside. Solving this is outside the scope of this study, but is important to look into if/when an import of FKB building data is on the agenda.

Another problem that arose when testing 3D modeling in OSM is overlapping nodes with different heights. 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 7.4a and how the building actually should have been modeled is shown in figure 7.4b. 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. This is something to consider if/when a new conversion script is implemented.



(a) Building who share nodes placed at the same position but have different heights



(b) Building in 3d.kommunekart.com

Figure 7.4: 3D representation of a FKB building, showing the problem with overlapping points with different heights in OSM

The last challenging problem that emerged through the 3D modeling part was that the building outline is not connected with the rest of the building lines in the SOSI representation. In SOSI, the area with the building feature type are only connected with the way representation creating the outline and there is no connection with the rest of the roof lines who creates the shape of the roof. When modeling the 3D buildings in this study, a manual search had to be done to find every line representing the roof of one building. One solution on how to collect every line that models one building was using the ..KP information in SOSI. In SOSI, every point that also is present in another way or area representation is marked with ..KP, where KP means junction (*knutepunkt*). Finding every line representing for one building by searching for every junction point was a time-consuming job. This has to be solved automatic through a script, that can, for instance, introduce a mutual reference id between all lines creating one building. Solving this is also outside the scope of this study.

The five most common building types are covering a large percentage of all building types as shown in figure 7.5 of the twenty most common building types in Trondheim. A good start is to look at a conversion script which converts the five most common types from SOSI into a working 3D representation in OSM. When or if that is achieved, the script can be adapted to the rest of the building types.

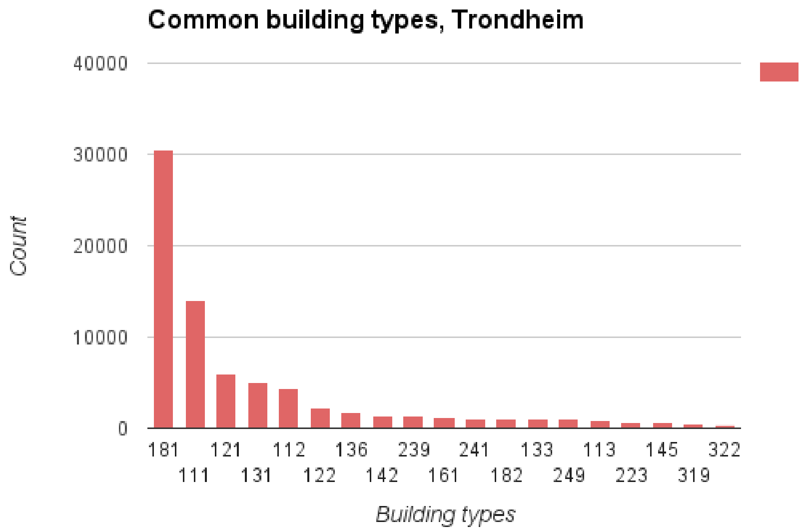


Figure 7.5: 20 most common building types in Trondheim, see table 10.1 for building type conversion. Source: PostGIS query in the cadastre

8 Discussion

The findings in this study show how the FKB building data could be imported into OpenStreetMap. The study introduces which method to use and also technical challenges and suggestions. The micro-tasking method is the suggested method for this study, and by evaluating other import methods and also imports that used this method, try to conclude if this is the preferred method when importing external data into OSM. This study hopefully also present good arguments in favor of importing FKB data into OSM.

Why import the FKB data? The FKB data is more accurate than for instance the TIGER data, and also the N50 data. The accuracy of FKB building data varies from +/- 0.20 m to +/- 2 m, depending on the FKB standard covered in the area the building lies in, as introduced in section 4.2. The N50 data is adapted to a scale range of 1:25000 to 1:100000 [Geonorge,], while FKB is adapted to a scale range of 1:5000 to 1:30000 [Geonorge, 2014]. The FKB data, especially the building data, can enhance the quality of the OSM-map. Improving the quality of data in OSM can increase the usage of the map. The community wants the map to be used, and if this can provide increased usage, the import will contribute in a positive manner. There will always be people negative to such an import, and there are good arguments on both sides. This study chose to see the arguments in favor of this import, especially if it results in a 3D model of all buildings in Norway. A free and open solution with a 3D model of all buildings in Norway will, with little doubt, increase the popularity of OpenStreetMap, in particular among the Norwegian people.

This paper only provides an overview study of how FKB buildings could be 3D modeled in OpenStreetMap. More time needs to be spent investigating possible methods. It is not certain that it's possible to find a uniform method for all building types. This study can't answer that. Further studies must be conducted to be able to evaluate how to model different building types and also evaluate different 3D modeling schemas supported by OpenStreetMap. A good start is, as mentioned, to look at the five most common building types and continue from there.

Another argument in favor of importing the FKB building dataset into OSM can be seen in the aftermath of the New York building import. 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. The solution was to offer the New York GIS department a subscription to daily email notifications on building and address changes in OSM. An excellent example of how government and open source can take advantage of each other. Such a collaboration could also be evaluated and discussed if/when an FKB import is approved.

No import method used by the OSM community today can make an import of existing data into the OSM database simple. It will always be challenging, and it will always require experienced OSM users. The scope of this study is evaluating if the micro-tasking method can help the community during imports of external data. As argued in this paper, the results are that the micro-tasking method helps the community. It makes the import easier to organize. The method also simplifies the import by dividing it into smaller parts, making each import task less complicated. By making each task less complicated, more people can contribute during the import by completing one task at a time. Many of the technical challenges are still there and new technological challenges arise. It is still necessary to convert the datasets to the XML file format used by OSM, and the method requires the dataset divided into tasks small enough to be finished within a reasonable time. It is important to be aware of these technical challenges.

9 Conclusion

If or when the community gets approval to import the FKB datasets into OpenStreetMap, the micro-tasking method should be the desired method to use. There are two building imports, in New York and Los Angeles, that used this method and both imports are evaluated as successful imports in this study. When doing an import, it is important to spend time on investigating existing tools before developing new ones. Many helpful tools have been developed already, both during the NY and LA building import. This will save the volunteers planning the import much time, especially if the micro-tasking method is used.

Results of this study show that micro-tasking is a suitable method for importing external data into OSM. It is not only a suitable method, this paper will conclude that it's the desired method in the OSM community for this kind of process, especially because of the well-developed tool, the OSM Tasking Manager. The tool and method make the import process easier for both the experienced user but even more important for the more amateur users. The experienced users can easily spread the information about the import using the OSM Tasking Manager web page, and amateur users can access necessary information much easier. Being aware of the technological challenges the micro-tasking method brings, this study still concludes that it is the best method for importing existing data into the OSM database today.

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 Date accessed: 2016-11-02.

[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 Date accessed: 2016-11-02.

[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/ Date accessed: 2016-12-02.

[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 Date accessed: 2016-10-27.

[Building, 2016] Building, L. (2016). List of usernames participating in the import. https://github.com/osmlab/labuildings/issues/40 Date accessed: 2016-12-16.

[Buildings, 2016a] Buildings, O. (2016a). Trondheim (NO) • OSM Buildings. http://osmbuildings.org/?lat=63.42411&lon=10.38373&zoom=18.8&rotation=0&tilt=30 Date accessed: 2016-12-20.

[Buildings, 2016b] Buildings, O. (2016b). Trondheim (NO) • OSM Buildings. http://osmbuildings.org/?lat=63.42687&lon=10.39787&zoom=19.6&rotation=0&tilt=30 Date accessed: 2016-12-08.

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

[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, 2016] edits OpenStreetMap, A. (2016). Automated edits - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Automated_edits&oldid=1319652 Date accessed: 2016-10-26.

[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/ Date accessed: 2016-10-21.

[Exel van, 2013] Exel van, M. (2013). State of the Map US: San Francisco 2013 - MapRoulette, one year later. https://vimeo.com/68098505 Date accessed: 2016-11-24.

[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 Date accessed: 2016-11-14.

[Geonorge,] Geonorge. N50 Kartdata - Kartverket - Kartkatalogen. https://kartkatalog.geonorge.no/metadata/kartverket/n50-kartdata/ea192681-d039-42ec-b1bc-f3ce0 Date accessed: 2016-12-08.

[Geonorge, 2014] Geonorge (2014). Felles KartdataBase (FKB) - Kartverket - Kartkatalogen. https://kartkatalog.geonorge.no/metadata/geovekst/felles-kartdatabase-fkb/0e90ca7 Date accessed: 2016-12-08.

[Gnonthgol, 2013] Gnonthgol (2013). [NUUG kart] sosi2osm. https://lists.nuug.no/pipermail/kart/2013-October/004314.html Date accessed: 2016-11-08.

[Goodchild, 2007] Goodchild, M. F. (2007). Citizens as sensors: the world of volunteered geography.

[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 Date accessed: 2016-10-31.

[Hansen, 2007a] Hansen, D. (2007a). [OSM-talk] TIGER upload queue (maps of entire US). https://lists.openstreetmap.org/pipermail/talk/2007-September/017621.html Date accessed: 2016-10-26.

[Hansen, 2007b] Hansen, D. (2007b). [OSM-talk] TIGER upload queue (maps of entire US). https://lists.openstreetmap.org/pipermail/talk/2007-September/017744.html Date accessed: 2016-10-27.

[HOT Tasking Manager, 2016a] HOT Tasking Manager (2016a). HOT Tasking Manager - #2261 - Tanzania Development Trust: Serengeti district mapping project part 2. http://tasks.hotosm.org/project/2261 Date accessed: 2016-12-16.

[HOT Tasking Manager, 2016b] HOT Tasking Manager (2016b). HOT Tasking Manager - #2261 - Tanzania Development Trust: Serengeti district mapping project part 2. http://tasks.hotosm.org/project/2261 Date accessed: 2016-11-21.

[Jørgenrud, 2013] Jørgenrud, M. (2013). Kartskatt til alle - Digi.no. http://www.digi.no/artikler/kartskatt-til-alle/286539 Date accessed: 2016-11-25.

[Kartverket, a] Kartverket. Om Geovekst-samarbeidet | Kartverket. http://www.kartverket.no/geodataarbeid/geovekst/Om-Geovekst-samarbeidet/ Date accessed: 2016-11-19.

[Kartverket, b] Kartverket. SOSI | Kartverket. http://www.kartverket.no/geodataarbeid/standarder/sosi/ Date accessed: 2016-11-11.

[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–innforing-i-kommunene/ Date accessed: 2016-11-09.

[Kartverket, 2016b] Kartverket (2016b). SOSI-standard del 2 Generell objektkatalog. http://www.kartverket.no/geodataarbeid/standarder/sosi/sosi-standarden-del-2/ Date accessed: 2016-11-07.

[Kartverket, c] Kartverket, S.-s. Bygningstype - Geonorge objektregister. objektkatalog.geonorge.no/Objekttype/Index/EAID_7C268089_ED2E_494b_9982_FD19A55B8F07#/uml Date accessed: 2016-11-20.

[Kartverket, d] Kartverket, S.-s. TreDNivå - Geonorge objektregister. objektkatalog.geonorge.no/Objekttype/Index/EAID_1DA8CD10_FAF4_4d0f_A46B_BB5D8ACF13E9 Date accessed: 2016-11-09.

[Kartverket, e] Kartverket, S.-s. Veranda - Geonorge objektregister. objektkatalog.geonorge.no/Objekttype/Index/EAID_3C31BCFA_6CE5_4af1_BB76_F7D2CAFD8FA8 Date accessed: 2016-11-11.

[Kihle, 2014] Kihle, K. (2014). [NUUG kart] Bruk av datasett fra Kartverket i OpenStreetMap. https://lists.nuug.no/pipermail/kart/2014-August/004831.html Date accessed: 2016-10-25.

[Lidman, 2014] Lidman, A. (2014). ICCM and micro-tasking tools in OpenStreetMap | Mapbox. https://www.mapbox.com/blog/international-conference-of-crisis-mappers-2014/ Date accessed: 2016-11-24.

[Los Angeles building, 2016] Los Angeles building, O. (2016). OSM Tasking Manager - #20 - Southeast - LABuildings Import. http://labuildingsimport.com/project/20 Date accessed: 2016-12-16.

[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/ Date accessed: 2016-11-14.

[McConchie, 2014] McConchie, A. (2014). How to divide up the tasks? #4 Github issues, LA buildings. https://github.com/osmlab/labuildings/issues/4 Date accessed: 2016-11-06.

[Mehus, 2014] Mehus, T. (2014). [Imports] N50 imports from Kartverket (The Norwegian Mapping Authority). https://lists.openstreetmap.org/pipermail/imports/2014-June/003252.html Date accessed: 2016-10-25.

[Mielczarek, 2007] Mielczarek, T. (2007). [OSM-talk] [OSM-dev] TIGER, which states next? https://lists.openstreetmap.org/pipermail/talk/2007-October/019231.html Date accessed: 2016-10-27.

[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.

[NORKART, 2016] NORKART (2016). Kommunekart 3D. http://3d.kommunekart.com/Clients/index.html?funksjon=vispunkt&y=269674.3363122087&x=7041000.767160203&z=115.8055861486594&lookatx=7041072.04808862&lookaty=269707.5914191979&lookatz=54.05506304857936&zoom=13&srid=32633&kommune=1601 Date accessed: 2016-.

[ODbL,] ODbL. Open Database License (ODbL) v1.0 | Open Data Commons. http://opendatacommons.org/licenses/odbl/1.0/ Date accessed: 2016-10-18.

[OpenStreetMap, 2016a] OpenStreetMap (2016a). OpenStreetMap. http://www.openstreetmap.org/#map=19/63.43014/10.39827 Date accessed: 2016-12-06.

[OpenStreetMap,] OpenStreetMap, C. LearnOSM. http://learnosm.org/en/osm-data/file-formats/ Date accessed: 2016-11-26.

[OpenStreetMap, 2007] OpenStreetMap, C. (2007). TIGER - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/TIGER Date accessed: 2016-10-21.

[OpenStreetMap, 2013] OpenStreetMap, C. (2013). Simple 3D buildings - One year experiences / OpenStreetMap 3D / OpenStreetMap Forum. https://forum.openstreetmap.org/viewtopic.php?id=19729 Date accessed: 2016-11-26.

[OpenStreetMap, 2014] OpenStreetMap, C. (2014). Import/Catalogue/N50 import (Norway) - OpenStreetMap Wiki. http://wiki.openstreetmap.org/wiki/Import/Catalogue/ N50_import_(Norway) Date accessed: 2016-10-25.

[OpenStreetMap, 2015a] OpenStreetMap, C. (2015a). Node - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Node&oldid=1189910 Date accessed: 2016-11-20.

[OpenStreetMap, 2015b] OpenStreetMap, C. (2015b). Node - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Node&oldid=1189910 Date accessed: 2016-12-01.

[OpenStreetMap, 2015c] OpenStreetMap, C. (2015c). ProposedRoofLines - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title= ProposedRoofLines&oldid=1193866 Date accessed: 2016-12-08.

[OpenStreetMap, 2015d] OpenStreetMap, C. (2015d). Simple 3D Buildings/Example Models - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Simple_3D_Buildings/ Example_Models&oldid=1194811 Date accessed: 2016-12-08.

[OpenStreetMap, 2016b] OpenStreetMap, C. (2016b). Automated Edits code of conduct - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Automated_Edits_code_of_conduct& oldid=1351097 Date accessed: 2016-10-26.

[OpenStreetMap, 2016c] OpenStreetMap, C. (2016c). building | Keys | OpenStreetMap Taginfo. http://taginfo.osm.org/keys/building#values Date accessed: 2016-11-21.

[OpenStreetMap, 2016d] OpenStreetMap, C. (2016d). Changeset - OpenStreetMap Wiki. https://wiki.openstreetmap.org/w/index.php?title=Changeset&oldid=1345995 Date accessed: 2016-11-21.

[OpenStreetMap, 2016e] OpenStreetMap, C. (2016e). Data working group - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Data_working_group&oldid=1404356 Date accessed: 2016-10-25.

[OpenStreetMap, 2016f] OpenStreetMap, C. (2016f). Elements - OpenStreetMap Wiki. https://wiki.openstreetmap.org/w/index.php?title=Elements&oldid=1339476 Date accessed: 2016-11-21.

[OpenStreetMap, 2016g] OpenStreetMap, C. (2016g). Import/Catalogue/Topography import for Norway - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Import/Catalogue/Topography_import_for_Norway&oldid=1342240 Date accessed: 2016-12-07.

[OpenStreetMap, 2016h] OpenStreetMap, C. (2016h). Import/Guidelines - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Import/Guidelines&oldid=1396104 Date accessed: 2016-12-19.

[OpenStreetMap, 2016i] OpenStreetMap, C. (2016i). MapRoulette - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=MapRoulette&oldid=1280971 Date accessed: 2016-11-24.

[OpenStreetMap, 2016j] OpenStreetMap, C. (2016j). Relation - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Relation&oldid=1318323 Date accessed: 2016-11-21.

[OpenStreetMap, 2016k] OpenStreetMap, C. (2016k). Simple 3D buildings - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Simple_3D_buildings&oldid=1389188 Date accessed: 2016-11-26.

[OpenStreetMap, 2016l] OpenStreetMap, C. (2016l). State Of The Map - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=State_Of_The_Map&oldid=1372231 Date accessed: 2016-11-21.

[OpenStreetMap, 2016m] OpenStreetMap, C. (2016m). Way - OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Way&oldid=1293953 Date accessed: 2016-11-20.

[OSM Tasking Manager,] OSM Tasking Manager, L. B. OSM Tasking Manager - LA buildings import. http://labuildingsimport.com/ Date accessed: 2016-10-31.

[OSMF,] OSMF. About | OpenStreetMap Blog. https://blog.openstreetmap.org/about/ Date accessed: 2016-10-18.

[OSMstats, 2016] OSMstats (2016). OSMstats - Statistics of the free wiki world map. http://osmstats.neis-one.org/ Date accessed: 2016-11-26.

[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/ Date accessed: 2016-11-14.

[Sambale, 2016] Sambale, M. (2016). OpenStreetMap | manings sin dagbok | Fixing split buildings in LA. http://www.openstreetmap.org/user/manings/diary/39230 Date accessed: 2016-11-06.

[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 Date accessed: 2016-11-25.

[SOSI-sekretariatet, a] SOSI-sekretariatet. BygningstypeKode - Geonorge objektregister. https://objektkatalog.geonorge.no/Objekttype/Index/EAID_A408BF17_3017_41c Date accessed: 2016-11-26.

[SOSI-sekretariatet, b] SOSI-sekretariatet. Takriss - Geonorge objektregister. https://objektkatalog.geonorge.no/Objekttype/Index/EAID_C8451D92_8DC5_4c38_A09E_3DD Date accessed: 2016-12-07.

[TagInfo, 2016] TagInfo (2016). building | Keys | OpenStreetMap Taginfo. http://taginfo.openstreetmap.org/keys/building Date accessed: 2016-11-29.

[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.

[Weber, 2008] Weber, P. (2008). User-Generated Street Maps. pages 12–18.

[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 Date accessed: 2016-10-21.

[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 Date accessed: 2016-10-21.

[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.

10 Appendix

10.1 FKB building type mapped to OSM building value

FKB: building type	count	OSM: building=*
181 - Garasje	30451	Garage
111 - Enebolig	13989	Detached
121 - Del av tomansbolig	6022	House
131 - Del av rekkehus	5143	House
112 - Enebolig	4351	Detached
122 - Tomannsbolig	2301	House
136 - Småhus	1705	House
142 - Boligbygg (3-4 etg)	1482	Appartment
239 - Lagerbygning	1380	Warehouse
161 - Fritidsbygg	1301	Cabin
241 - Hus for dyr	1111	Farm_auxiliary
182 - Garasje	1060	Garage
133 - Atriumshus	1028	House
249 - Landbruksbygning	1007	Agricultural
113 - Våningshus	796	Farm
223 - Transformator	750	Substation
145 - Boligbygg (3-4 etg)	741	Appartment
319 - Kontorbygning	514	Office
322 - Butikk/forretningsbygg	365	Commercial

Table 10.1: Mapping FKB buildingtypes to OpenStreetMap values, part 1

143 - Boligbygg (5 el. fler etg.)	310	Appartment
231 - Lagerhall	309	Warehouse
329 - Forretningsbygning	303	Commercial
199 - Annen boligb.	300	Yes
141 - Boligbygg (2 etg.)	290	Detached
612 - Barnehage	255	Kindergarten
183 - Naust, båthus, sjøbu	252	Boathouse
152 - Studentboliger	238	Dormitory
146 - Boligbygg (5 etg.)	218	Appartment
219 - Industribygning	213	Industrial
212 - Verkstedbygning	197	Service
619 - Skolebygning	173	School
144 - Boligbygg (2 etg.)	162	Apartment
243 - Veksthus	160	Greenhouse
613 - Barneskole	135	School
659 - Idrettsbygning	125	Sports_hall
123 - Del av våningshus, bolighus m/ to boliger	112	House
135 - Terrassehus	97	Apartment
211 - Fabrikkbygning	95	Factory
531 - Restaurantbygning	88	Restaurant
311 - Kontorbygning	83	Office

Table 10.2: Mapping buildingtypes to OpenStreetMap values, part 2

10.2 Code snippet from .lua file, setting building=* value

Listing 10.1: 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" then
        out["building"] = "detached"
    elseif tokens[2] == "121" or tokens[2] == "123" or tokens[2] == "131"
    or tokens[2] == "122" or tokens[2] == "133" or tokens[2] == "136" then
        out["building"] = "house"
```

```
elseif tokens[2] == "135" or tokens[2] == "142" or tokens[2] == "143"
or tokens[2] == "144" or tokens[2] == "145" or tokens[2] == "146" then
    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
```

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

Listing 10.2: 3D representation of a house from FKB data in OSM XML

```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" />
```

```xml
        <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>
```