HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

COMPUTER SCIENCE

ICOM
INSTITUTE FOR
COMMUNICATION SYSTEMS

# Preparation of 3D Data for HSR-Game with Motion

## Term Project

Department of Computer Science

University of Applied Science Rapperswil

Autumn Term 2016


Author(s): Sophie Somerton, Joel Hochreutener

Advisor: Prof. Stefan Keller, Geometa Lab HSR

Project Partner: ICOM, Institute for Communication Systems

# Edition Notice

## Location

Rapperswil, Switzerland

## Period

Autumn 2016

## Version

Version 3.0 – 07.01.17

## License



## Change History

| Datum | Version | Change | Author |
|---|---|---|---|
| 21.10.2016 | 1.0 | Title page and structure | Sophie Somerton |
| 29.11.2016 | 1.1 | Main document completed | Sophie Somerton |
| 12.12.2016 | 2.0 | New structure | Sophie Somerton |
| 23.12.2016 | 2.1 | First release | Sophie Somerton |
| 07.01.2017 | 3.0 | Final version | Sophie Somerton |

# Abstract

In this project different methods of creating a 3D model of an area with several buildings were evaluated. Aim was to create a realistic environment for a game. The workflow of the different methods should be transferable, meaning that other users can easily adapt the workflow and use it for their area. The evaluation was focused on two main approaches: The first is using purely open data and open source software, such as OSM and OSM2World. The other approach uses commercial photogrammetry software with and without the help of a drone.

As expected using photogrammetry creates a much more realistic model than building one out of OSM data. On the other hand it can cost around 5000 CHF if you have to buy the involved software and tools (camera/drone). We assume that some universities may already have these tools and software in-house. Either way it takes a considerate amount of time to create a realistic 3D model of buildings. But in the end both methods archive a usable 3D model. Depending on how much time, available tools and preferred approach one or the other method will be more suitable.

All documents and results of this project can be found at following link:
https://github.com/cjhox/VRmotion_HSR-Game

# Management Summary

## Initial Situation

As an initial situation there is an electrical engineering institute (ICOM) which owns two 6DOF Motion-Simulators. It is ready to play integrated games and use the Unreal Engine which combines motion-data (from any object) with the simulators. Next to that there exists OpenStreetMap (OSM) data which can be used also for 3D scenes. For example the research centre of HSR campus is mapped in a very detailed way. What is missing there and in OSM in general is the texturing. As an alternative approach to that, there are very powerful photogrammetry tools like Agisoft PhotoScan or Pix4D. Therefore, the focus of this thesis was to evaluate these two approaches: The first is using purely open vector data, free textures and open source software, such as OSM and OSM2World. The other approach uses picture data, point cloud data and commercial photogrammetry software with and without the help of a drone.

## Approaches and Technologies

To estimate the workload of the approach using OSM2World, it was decided to perform a short test run. By means of this prototype it became evident that a model without any texturing can be generated in practically no time. The progress of the implementation of the first approach (the 2.5D data-driven approach 1 using OSM) was faster than expected.

Given that additional time it was decided to spilt the photogrammetry approach into two parts: primo a manual approach with camera and commercial software (Agisoft PhotoScan) and secondo using a drone with another commercial software (Pix4D). These (sub-) approaches using Agisoft PhotoScan (2a) and Pix4D (2b) have no dependencies and could be evaluated and documented separately.

Due to that the results were to be used in a follow-up project the model needed to be compatible with a game engine. The last task was to ensure that the model can be integrated into a game engine like Unreal Engine or Unity.

## Results

The three approaches 1, 2a and 2b have been evaluated. Both main approaches (1 and 2) could be completed and created a 3D model of the HSR research centre building. The sub-approach 2a, taking the pictures manually with a camera, didn't work as well as anticipated. The other approaches worked very well. A workflow for all approaches was established and documented.

As expected using photogrammetry creates a more realistic and detailed model than the one out of OSM data. On the other hand, the photogrammetry approach is more expensive with all its

software, tools and hardware (camera/drone) involved. Either way, it takes a considerable amount of time to create a realistic 3D model of buildings.

In the end both approaches (1 and 2) produce a 3D model which is fit for use for the purposes of a game. To take the photos for the photogrammetry approach (2) it is more convenient to use a drone than to take the pictures manually. Depending on the available time, tools and the requirements towards the model, one or the other approach is more suitable.

| | **OSM2World Approach (1)** | **Photogrammetry Approach (2)** |
|---|---|---|
| **Additional equipment needed** | - | reflex camera or drone |
| **Licensing cost** | free | 185 CHF - 3590 CHF (Agisoft)<br>3200 CHF - 7900 CHF (Pix4D) |
| **Approx. expenditure of time (worst case)** | 70 hours | 60 hours (reflex camera)<br>40 hours (drone) |
| **Approx. expenditure of time (best case)** | 40 hours | 25 hours |
| **Suitability of the result, strong points** | • without texturing very quick generated<br>• sharp edges<br>each object is represented as a separate object, editable<br>• suited for ground level near object model | • faster to create<br>• able to depict any object<br>• very realistic<br>• great for aerial overview models |
| **Expandability of area** | Whole workflow needed to be done again, but time consuming work points such as texturing are reduced. | Easy to add a new area part, may need to add additional tie points (time consuming). |

# Outlook

The results of this project are the basis for a follow-up project called “Interactive Multiplayer HSR-Game with Motion”. Two 3D models are available now as a starting point, ready to be integrated into a game engine.

The topic of generating 3D models is wide spread. As a consequence this thesis focused on only two approaches. Another approach would be using a laser scanner. Further uncovered topics include indoor mapping, geometry as model base and other photogrammetry software.

# Contents

# 1 Introduction

## 1.1 Project Overview

The subject of this term project is 3D mapping of buildings. The aim is to prepare approach specific data, generating 3D models and evaluate the feasibility of a game based on a generated model using a game engine such as Unreal Engine or Unity. One of these models shall later be used as environment for a HSR-game.

## 1.2 Initial Situation

As an initial situation there is an electrical engineering institute (ICOM) which owns two 6DOF Motion-Simulators. It is ready to play integrated games and use the Unreal Engine which combines motion-data (from any object) with the simulators. Next to that there exists OpenStreetMap (OSM) data which can be used also for 3D scenes. For example the research centre of HSR campus is mapped in a very detailed way. What is missing there and in OSM in general is the texturing. As an alternative approach to that, there are very powerful photogrammetry tools like Agisoft PhotoScan or Pix4D. Therefore, the focus of this thesis was to evaluate these two approaches: The first is using purely open vector data, free textures and open source software, such as OSM and OSM2World. The other approach uses picture data, point cloud data and commercial photogrammetry software with and without the help of a drone.

## 1.3 Project Aim

As mentioned above, this project shall be the base for a further project to create a game for the 6DOF Motion-Simulators. Primary targets are an evaluation of the possibilities to create a 3D model of an existing building and instructions to this process. The secondary target is to create a textured and detailed 3D model of the HSR research centre which is usable in a game engine.

## 1.4 Motivation

Publicity is important at a university. Especially for one which has a good name in IT. At exhibitions or when groups visit the school it is important to have an interesting exhibit piece. The 6DOF Motion-Simulators are a very good eye-catcher. Next step is to combine that eye-catcher with an association to HSR. Possible ideas are a flight or rollercoaster ride through the campus or even to be able to drive around. All these ideas have in common that a 3D model of the HSR campus is needed. To be able to create a cool racing game a detailed 3D model of the environment is necessary.

# 2 Prototype: Estimation of Workload Using OSM and OSM2World

The Prototype shall give an idea how much effort creating a 3D model involves. Next to that it will show how detailed the data from OSM is.

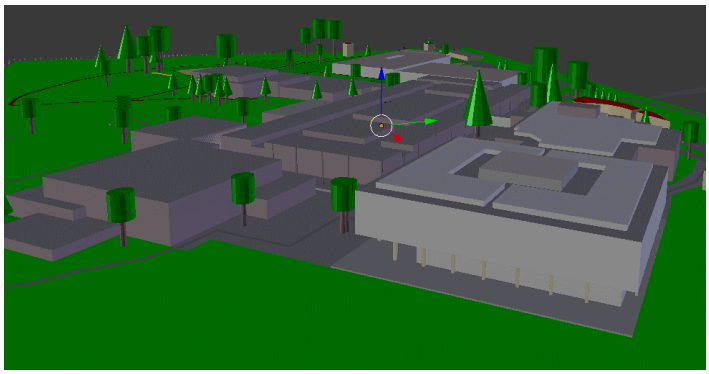

**Figure 1: Overview of the campus**

## 2.1 Insights

With the help of the prototype it was possible to win some very interesting insights. Essentially all the buildings are mapped with great detail. Each building is easily recognisable. All the buildings are available and have the right structure. Still there are some mismatches to the real world. For example the heights of the buildings are wrong. More faults are the missing lake, tables, chairs, statue and umbrellas by building 8 and the benches mostly have the wrong form and orientation. Some of them are even on the wrong height level.

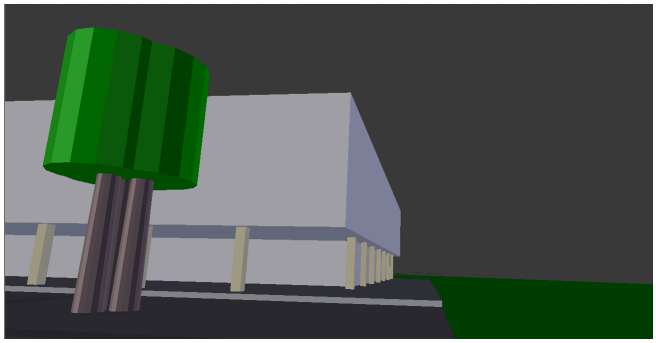

**Figure 2: Building 8 missing statue, table, chairs and umbrellas**

An important point is the texturing. Though the proper materials are tagged it lacks a good texturing. Meaning it is clear what material the area should be, but it is simply coloured in grey, green or brown.

## 2.2 First Texturing

The first experience with texturing were daunting. No chance to reach any kind of realistic looking textures. But it was possible to add some seamless textures for an acceptable result. Here some of the best perspectives from the first experiments with texturing:

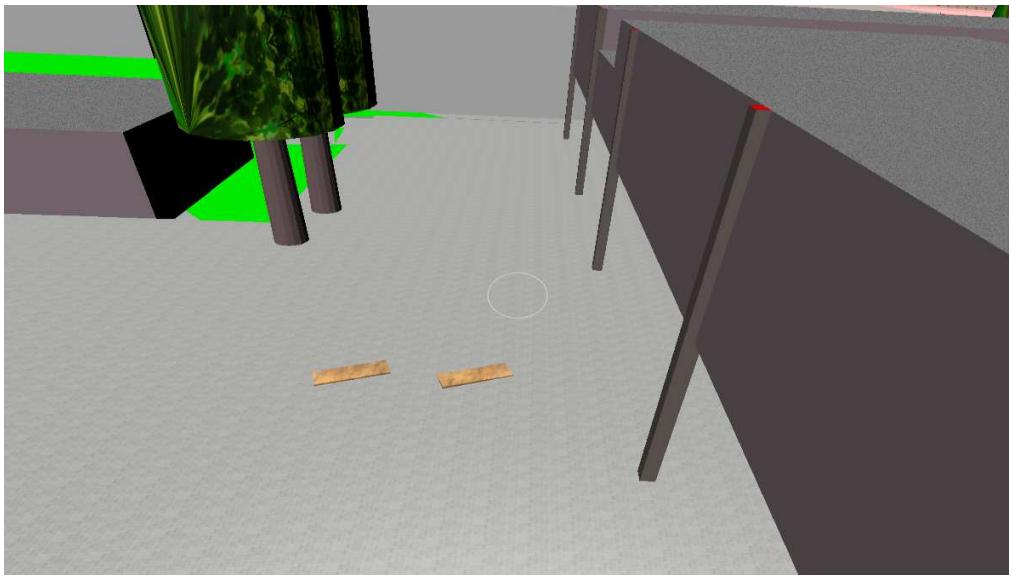

**Figure 3: Overview texture sample**

**Figure 4: Pathing stone**

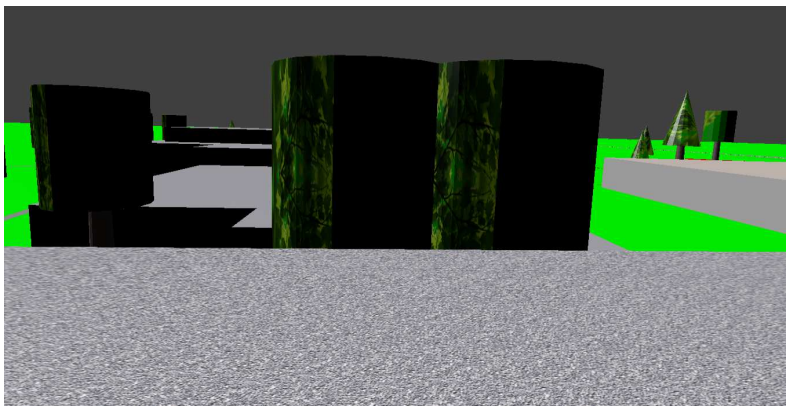

**Figure 5: Gravel and tree texture**

## 2.3 Integration

The next step in the prototype phase was to import a 3D model into Unreal Engine. There were some difficulties because the game engine threw a fatal error multiple times. Finally it was possible to fly around in the OSM data based 3D model by importing it via the FBX file.

## 2.4 Conclusion Prototype

It is fairly simple to create a 3D model out of the OSM data. The usefulness of the model depends strongly on how detailed the available OSM data is. An imprecise model can be won with a few clicks, but it will take some time to patch all the minor mistakes. The model can be exported and imported into a game engine. The question remains how realistic the model can be made using texturing. This shall be the focus in the following chapter “Evaluation”.

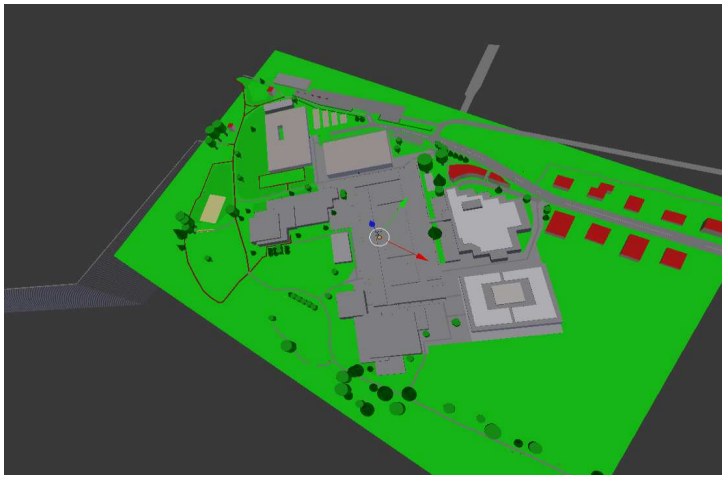

**Figure 6: Bird's-eye view**

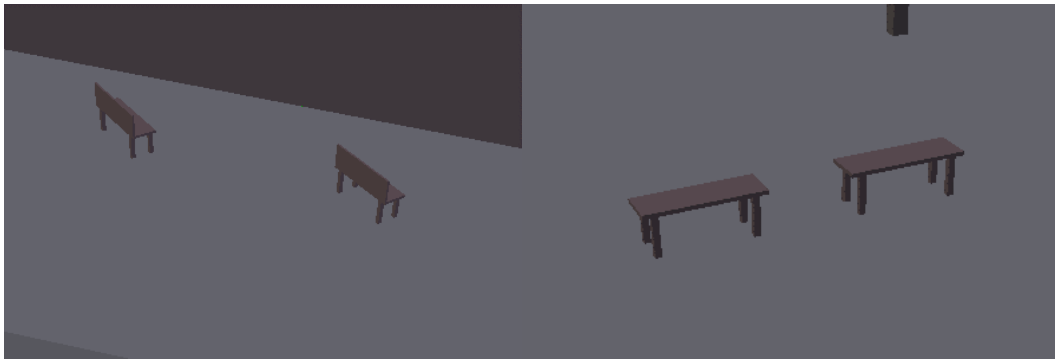

**Figure 7: Benches before and after adjustments (orientation and form)**

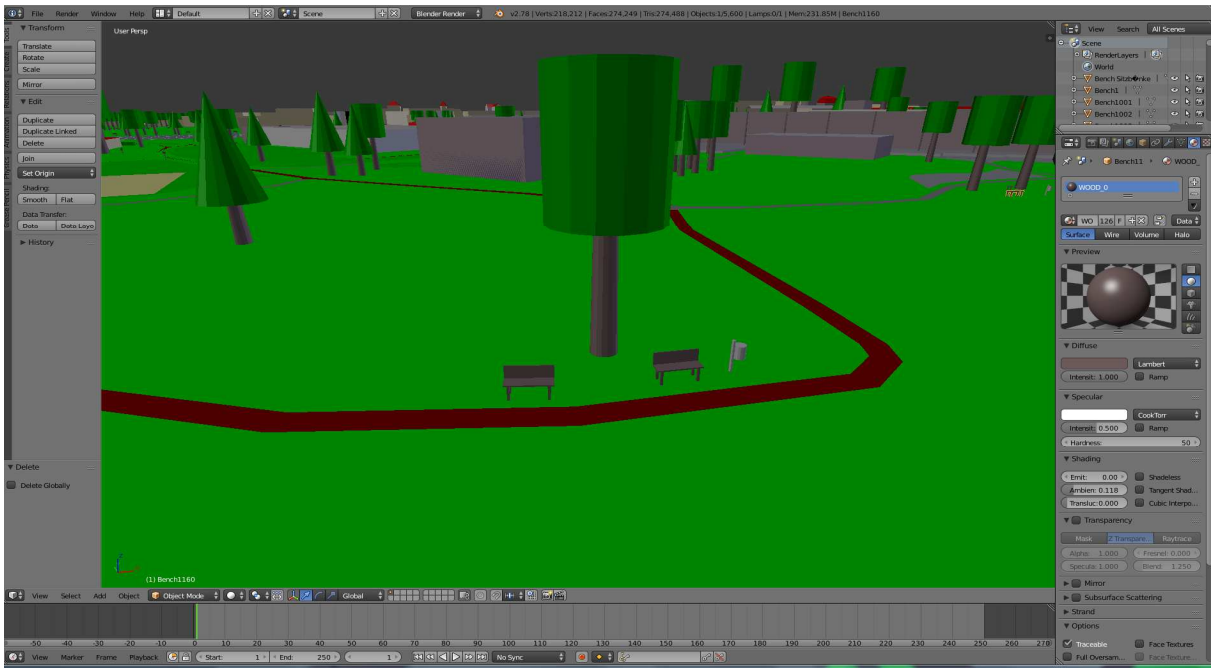

**Figure 8: Screenshot of Blender**

# 3 Evaluation: Basic Ways of Creating a 3D Model of Buildings

This evaluation contains different kinds of acquisition techniques for modelling buildings. There are 3 main ways: laser scanning, photogrammetry and manual measuring. Using software to design or represent a building is often called Building Information Modelling, short BIM. The focus of our evaluation lies on generating a 3D mesh out of OSM data (1), photogrammetry using Agisoft PhotoScan (2a) and photogrammetry using a drone with Pix4D (2b).

## 3.1 Workflow of Generating a Textured 3D Mesh Based on OpenStreetMap and OSM2World (Approach 1)

### 3.1.1 Introduction

The idea of this approach is to take open source data and open source software to generate a 3D model. By mapping the data in OSM you will lose lots of details, but you will get a 3D model with less bumps and distortions.

### 3.1.2 Requirements

The only requirement for generating a textured 3D mesh using OSM data is having a computer, all software are open source. The time you need to generate a model depends on how accurate it has to be and on how much data has already been mapped in OSM.

### 3.1.3 Tools

#### 3.1.3.1 OpenStreetMap (OSM)

*OpenStreetMap is a free, editable map of the whole world that is being built up and extended by volunteers and holds an open-content licence.*

*The OpenStreetMap licence allows free (or almost free) access to OSM images and all of its underlying map data. The project aims to promote new and interesting uses of this data*[1].

#### 3.1.3.2 OSM2World

*OSM2World is a converter written that creates three-dimensional models of the world from OSM data. The model can be exported to different formats.*[2]

#### 3.1.3.3 Blender

*Blender is the free and open source 3D creation suite. It supports the entirety of the 3D pipeline-modelling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation.*[3]

### 3.1.4 Workflow

The main tool for generating a 3D model out of OSM data is OSM2World. But this tool is not as powerful as you would need for well textured 3D models. This chapter will describe how to use OSM2World and how to extend its abilities.

[1] https://wiki.openstreetmap.org/wiki/About_OpenStreetMap
[2] https://wiki.openstreetmap.org/wiki/OSM2World
[3] https://www.blender.org/

#### 3.1.4.1 Extend OSM Data

First of all, you will have to search on the OSM webpage for the area you want to convert to a 3D model. After you found the area, check for missing objects or details and extend the OSM data with the online tool iD. iD is a html based editor embedded in the browser. Select the *Edit* register and sign up to edit the OSM data. There is a tutorial that will introduce you to using iD. There exist other editors, like JOSM, which have similar, less or even more capabilities for editing OSM data.

#### 3.1.4.2 Export Data

If the OSM data is as detailed as you want it to be, select the *Export* register at OSM main page and centre the needed area in the browser window. Pressing the *Export* button will export everything you see in the browser window. In the export section there is the possibility to set the export frame manually. When the frame is set correctly you can export the OSM file and save it to the computer.

#### 3.1.4.3 Check OSM File with JOSM

The export function of the OSM webpage will export only nodes and ways. Areas, which are partly outside the export boundaries, will not be exported completely. OSM2World cannot convert partly downloaded areas and therefore will ignore these.

To avoid this open the OSM file with JOSM and watch over each entry in the *Relations*-list at the bottom right corner. Incomplete objects are marked as incomplete. Right click on the objects you want to complete and choose the *Download incomplete members* command to download the missing parts. The routes and other way relations do not have to be complete, as it may not be suggestive to download whole street networks. Multi-polygons like buildings and other area relations should be complete.

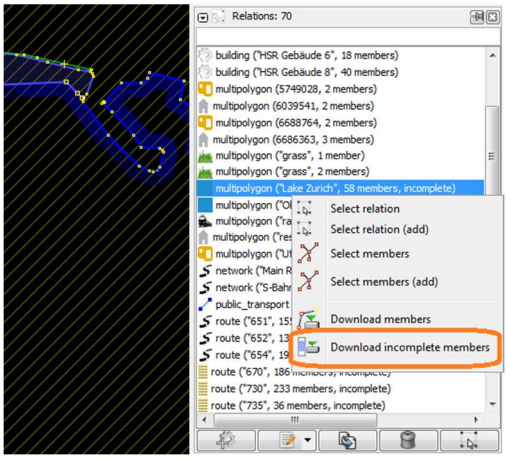


**Figure 9: JSOM relations-list**

After downloading the missing parts save the changes made to the OSM file, but do not export them to the OSM database.

#### 3.1.4.4 Convert OSM File with OSM2World to OBJ file

Start the OSM2World java program and open the OSM file using *File > Open OSM file*. At this state of process you can see what it will look like in 3D but without textures. It is difficult to navigate in the 3D viewer of OSM2World, so do not spend too much time there. Use *File > Export OBJ file* to convert the OSM file to an OBJ File and save it to the computer.

#### 3.1.4.5 Import the OBJ File into Blender

For the next step we will use Blender. If you are not used to Blender, work through some tutorials[4] first or try to follow these steps as exactly as possible. Shortcuts and instructions will be written at the end of the step. Pay attention to the position of your curser, some shortcuts will apply to the window you are hovering.

- Open Blender and delete the cube in the main window.
  - right click the cube to select it → "x" → enter
- Open the file browser for importing OBJ files.
  - Menu *File > Import > Wavefront (.obj)*
- Map the orientation of the OBJ file to *X Forward* and *Y Up*
  - In the menu on the left located in *Import OBJ*
  - Forward: X Forward
  - Up: Y Up
- Import the OBJ file
  - Double click the file.

#### 3.1.4.6 Implement Textures

A big advantage of OSM2World is its ability to assign the materials to the 3D objects. Adding textures to materials is easy. Just repeat the following instructions for each material and you will have a textured 3D model. Quality of the result depends on the quality of the chosen texture.

Select an object by right clicking it and go to the *Materials* register in the *Properties* window. Choose the material you want to add a texture and change to the *Texture* register.

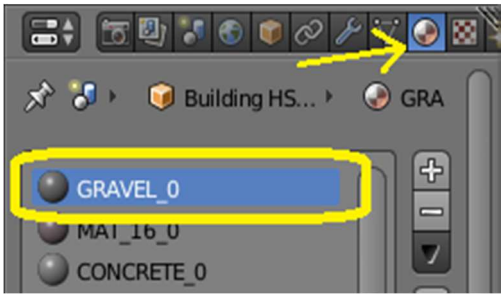


**Figure 10: Materials register**

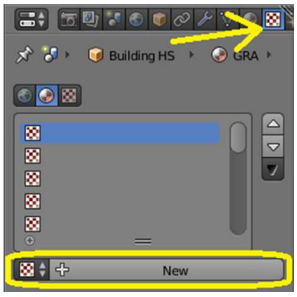


**Figure 11: Textures register**

Press the *New* button and choose the **Type of texture** you need. There are some calculated textures like the **cloud** or the **noise texture**, but normally you will need to import textures as images.

[4] We recommend http://blenderhilfe.de/?product=einfuehrung-in-blender


Figure 12: Texture type


Figure 13: Image section

On the webpage http://www.textures.com/ you can find lots of textures to download. To link the texture, press the *Open* button in the section *Image* and select your texture. Now you need to optimize the mapping of the image. If you are not used to the mapping methods, use **global** and a projection depending on the form of your objects.

#### 3.1.4.7 High Quality Textures

To create high quality textures watch the "Blender 3D Einsteiger Training" tutorial by blenderhilfe.de[5]. The important part is chapter C, where you can find information about texturing. You do not have to configure the shading, because the shading settings will not be exported into OBJ files.

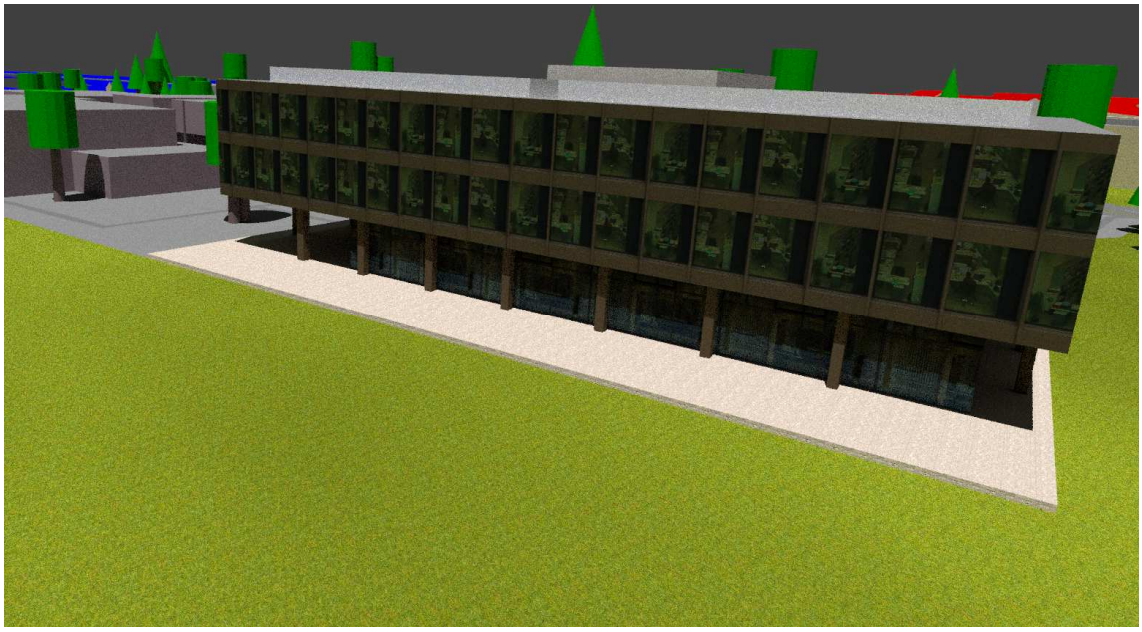

**Figure 14: Model with high quality texture**

[5] http://blenderhilfe.de/?product=einfuehrung-in-blender

### 3.1.5 Flowchart

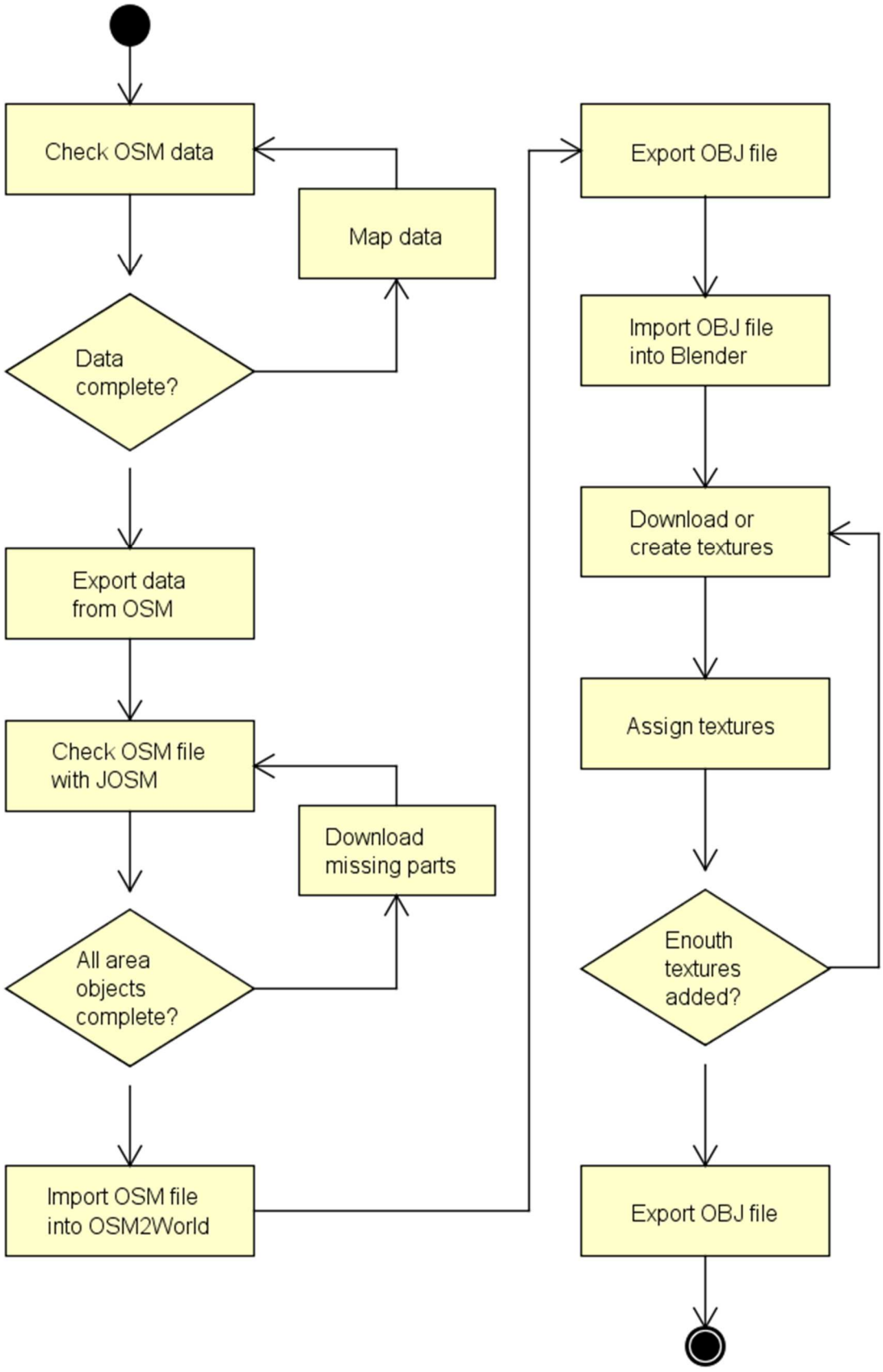


**Figure 15: Flowchart of the workflow with OSM**

### 3.1.6 Extend OSM2World

As a part of the evaluation, we analysed the tool OSM2World and tried to extend its functionality. It turned out that it is easier than expected.

#### 3.1.6.1 Source Code

The source code is located at https://github.com/tordanik/OSM2World and can easily be downloaded. There is an ant build file to simplify the build of your changes. Below a brief description of the project structure

**console**

Contains the commands and functions for using the jar file over the console.

**viewer**

Contains the menu commands and functions to preview the 3D model in the program window.

**core.osm**

Contains the functions to read OSM data and store them for further processing.

**core.hightmap**

Should contain the functions to read elevation information and store them, but not yet implemented.

**core.map_data**

Contains the classes and functions to store the OSM data in a more OSM2World specific way.

**core.map_elevation**

Contains the classes to store the elevation information for the map data and the functions to apply them.

**core.world**

Contains the modules to convert map data objects to a 3D objects. This package is responsible for handling the tags mapped in OSM.

**Core.target**

Contains the converter methods for the specific targets and the functions to save them.

#### 3.1.6.2 Extend OSM Tags

Some details you may want to figure in 3D cannot be mapped with tags defined by OSM wiki. OSM itself is able to carry any tag you want it to, but it is important, that you get the acceptance of the community before you define a new tag. This step you do not have to note if you just want to improve the representation of the objects.

#### 3.1.6.3 From Tag to Object

OSM data are represented as a collection of tags which OSM2World changes to a vector based 3D representation. Therefore for example a wastebasket will always look the same. You can

either add more property tags to the wastebasket to define its height, direction, colour or diameter and then extend OSM2World to handle these tags or you can just change its basic representation.

#### 3.1.6.4 Example: Table

This is a demonstration of how we extended the OSM2World with the ability to display tables.

On GitHub[6] you can find the commit containing all changes we made in the code.

##### Define a New Node Tag

As a first step, we defined “amenity=table” as a node tag for tables. It is important to add these definitions to the OSM wiki with all their property tags. But first we talked to some members of the community about the advantages and the details of the idea. Finally we added some tags to the OSM database to be able to test our code later on.

Here is the resulting wiki page: https://wiki.openstreetmap.org/wiki/Tag:amenity%3Dtable.

There is already a node tag for picnic tables which is not yet implemented in OSM2World. This allows us to cover two objects with one change.

##### Create a Table Class

The table fits best in the *StreetFurnitureModule*, which is why we created a new class named *Table*[7] there. Thanks to other classes like *Bench* we were able to copy some code snippets. At the top of the java file you can find the mapping of node tags and the representation class, what also needed to be completed.

##### Result

Finally we built the code and ran it with our OSM test file. Here is what the results looks like:

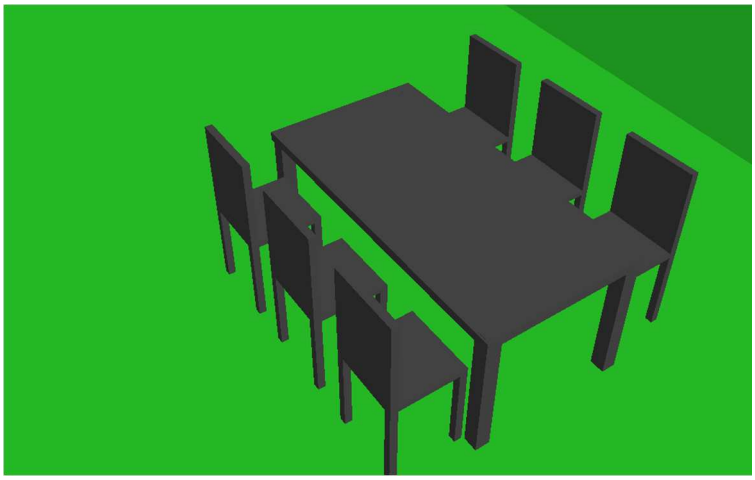

**Figure 16: Table generated by OSM2World**

[6] https://github.com/tordanik/OSM2World/commit/187efc06f1a15225cd8bb148c279a9ef7055969f
[7] Code can be found in the appendix.

### 3.1.7 Pros and Cons

#### 3.1.7.1 Advantages

- Each object is separate editable, therefore the mesh can be easily manipulated
- Large wiki website with helpful information exists
- Very quick if sufficient OSM data present
- Buildings can be mapped very detailed
- Low costs

#### 3.1.7.2 Downsides

- Accuracy cannot be guaranteed
- Faces are subdivided unnecessarily
- Calculation mistakes can happen
- Faces overlap
- Difficult to achieve a realistic texture

### 3.1.8 Problems

Some objects, like underpasses, are wrong implemented in OSM2World and need to be removed by hand. Sometimes there can be “phantom points” in the OSM file. These points can lead to mistakes in the model generated by OSM2World which need to be manually corrected. Next to these we have encountered no other problems with using OSM, OSM2World or Blender.

## 3.2 Workflow of Creating Textured 3D Meshes Using Agisoft PhotoScan and a Reflex Camera (Approach 2a)

Notice: The statements and principles of this chapter are based on the Agisoft PhotoScan user manual[8] and conversation with Marianne Deuber from Terradata[9].

### 3.2.1 Introduction

To get a photorealistic model generating it from map data and manually texturing will not suffice. This approach is based on photogrammetry or to be exact on stereophotogrammetry. By capturing a 3D object in pictures the depth is lost. If certain points are found on two or more photos the depth can be estimated.

### 3.2.2 Requirements

To build a model using photogrammetry certain requirements must be met. You will need a reflex camera with at least 10 megapixel resolution. Furthermore you will need the Agisoft PhotoScan software license and a reliable and powerful computer hardware.

The weather also has a great influence on the outcome. Rain may create reflections on flat surfaces. Sunny weather may cause reflections in windows. Optimal weather conditions would be light overcast.

### 3.2.3 Tools

#### 3.2.3.1 Agisoft PhotoScan

> *Agisoft PhotoScan is a stand-alone software product that performs photogrammetric processing of digital images and generates 3D spatial data.[10] It is capable of processing up to tens of thousands of photos yielding result products characterized by high degree of accuracy in both the horizontal and vertical dimensions.[11]*

#### 3.2.3.2 Camera

For our evaluation we used a Pentax K20D reflex camera. Basically any reflex camera with minimum of 10 megapixel resolution. Ideal would be a camera with geolocation function.

### 3.2.4 Workflow

Agisoft PhotoScan guides you quite nicely through the main workflow of generating the 3D mesh. You basically just have to follow the possible actions in the registry *Workflow*. But to get the most out of your model you need a profound knowledge about all the settings.

[8] http://www.agisoft.com/pdf/photoscan-pro_1_2_en.pdf
[9] Sitzungsprotokolle.docx, page 5
[10] http://www.agisoft.com/
[11] https://en.wikipedia.org/wiki/PhotoScan

Here a short estimation of expenditure of time based on a project with 1200 Photos and slightly above average computer hardware:

| Step of the process | Expenditure of time |
|---|---|
| Quality estimation of photos | 5 minutes |
| Alignment of the photos | 35 hours |
| Mesh based on sparse cloud | 2 minutes |
| Texturing of the mesh | 15 minutes |
| Build the dense cloud | 30 hours |
| Mesh based on dense cloud | 50 minutes |
| Texturing of the mesh based on dense cloud | 30 minutes |

#### 3.2.4.1 Taking the Pictures

When taking the pictures you must consider some points. First the autofocus should be turned off. When using a zoom object it is advised to fixate the zoom after focusing with some tape. While taking the pictures you should always keep the same distance. Also try to keep the same angle towards the front of the building. You should aim to achieve an overlapping of 80-90%. Each point on a picture should be found in at least 4 other pictures. It is better to have too many than too few pictures. To summarize: aim is to take many sharp pictures without changing any camera settings.

#### 3.2.4.2 Loading and Quality Estimation of Photographs

First of all, you will have to add all the pictures. This can be done in the registry *Workflow* by clicking on the button *Add Photos* or *Add Folder*. It is more like a reference to where the photos are located, you only indicate the photographs that will be used. All selected photos will appear in the photos work pane.

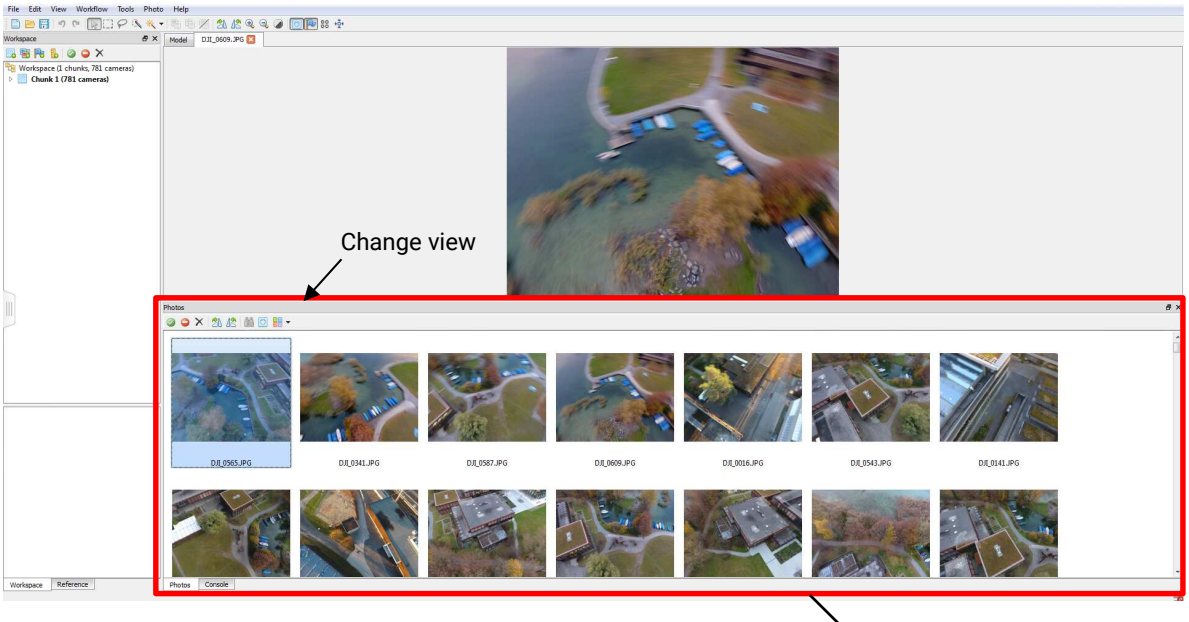


**Figure 17: Agisoft PhotoScan workspace**

To detect pictures of bad quality you can estimate the quality of the photos then right click on a picture and choose *Estimate Image Quality*. Now you can run the estimation on the chosen picture, all pictures or on the entire workspace. It takes about five to ten minutes to complete the estimation of a thousand pictures. To see the results you have to change the view of the photos work pane to *Details*. If the quality index is under 0.5 it is advised to remove or retake the photo.

#### 3.2.4.3 Masking Items on Images

There may be some unwanted items on the images such as people or shadows or maybe parts of the building are offset for example a railing. This may influence the texturing or even the mesh. To prevent this you can set a mask over those unwanted items. As setting masks is time consuming it is advised to use this only when needed.

To set a mask use the *rectangle selection* or *intelligent Scissors* to mark out the unwanted item. In the register *Photo* choose the command *Add selection.* The mask is now set.

#### 3.2.4.4 Chunks and Markers

If the building is similar on two sides, it is wise to set some markers. You should have at least 4 markers per picture. Less may suffice, but the more the better the result will be. The markers have to be placed as exact as possible, so make sure you zoom in properly. If the markers alone are not enough you can make chunks.

The recommendation from Agisoft PhotoScan is to create a chunk for each side of the building. You can add chunks by clicking on the *Add chunk* icon in the workspace. Select a chunk and add all the photos that show a part of for example the east face of a building as described above. Do the same for the other sides as well. The alignment of the photos must be done for each chunk separately. When the pictures are aligned it is possible to align and merge the chunks. For this select the *Align Chunks…* and then the *Merge Chunks…* command from the workflow register.

#### 3.2.4.5 Alignment of Photos

When all the needed preparation work is done, you can give the command to align the photos. This can be done by clicking in the workflow register on *Align Photos…*. Important parameters for the alignment are accuracy, pair preselection, point limit and the checkbox constrain features by mask.

##### Accuracy

The higher the accuracy is set the more accurate the camera position will be estimated. But it will also take much longer to fulfil the task. It is recommended to only use the highest accuracy setting with very sharp image data.

##### Pair Preselection

With a large set of photographs it can take a very long time to process. The process may be sped up when enabling pair preselection. In **Generic** preselection mode the overlapping pairs of photos

are selected by matching photos using lower accuracy setting first. If the camera locations have been measured the **Reference** preselection mode will be enabled. This mode bases the overlapping pairs of photos on the measured locations.

#### Key Point Limit

As an advanced setting it is possible to set an upper limit of feature points per image. Setting it to zero value allows the program to find as many key points as possible. It may also lead to a big number of less reliable points.

#### Tie Point Limit

It is also possible to set an upper limit on the number of matching points for every image. Using zero value does not apply any limit.

#### Constrain Features by Mask

If you have set some mask on the pictures you can enable *constrain features by mask*. This way all features detected in the masked areas are discarded.

#### Adaptive Camera Model Fitting

This option enables automatic selection of camera parameters to be included into adjustment. Especially when modelling buildings this setting helps to adjust more parameters during initial camera alignment. It will now take quite some time to generate the sparse point cloud.

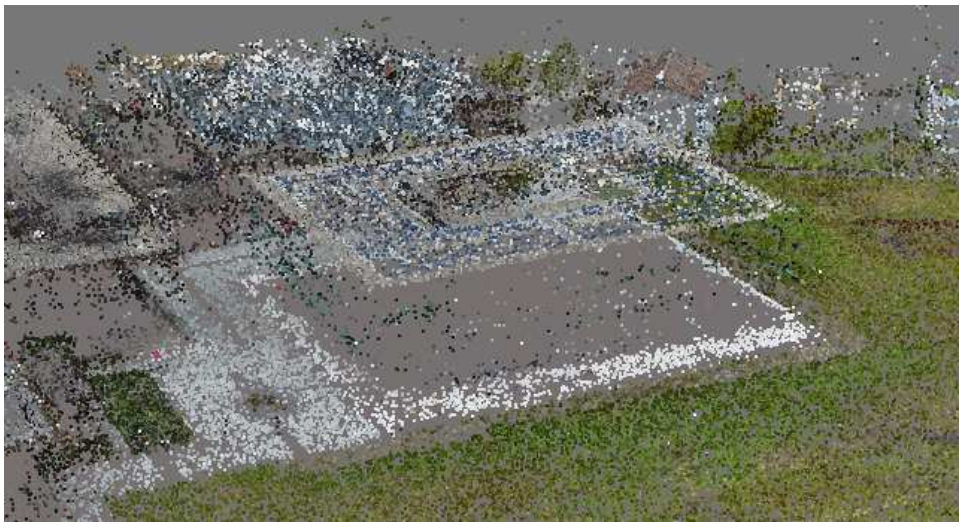

**Figure 18: Sparse point cloud**

### 3.2.4.6 Building the Dense Cloud and Mesh

PhotoScan is able to calculate depth information for each camera base on their positions. It is able to combine these information in a single dense point cloud. The density of this point cloud is comparable with LIDAR point clouds. To build the dense cloud select the *Build Dense Cloud* command from the workflow menu.

### Quality

Higher quality obtains a more detailed and accurate geometry, but requires longer time for processing.

### Depth Filtering Modes

Due to poor texture of some elements or badly focused images there can be some outliers among the points. PhotoScan offers different filtering algorithms depending on the project. **Mild** depth filtering mode is recommend if the geometry of the scene is complex with many details in the foreground. If the scene does not contain many meaningful details, then it is advised to use the **Aggressive** filtering mode. **Moderate** depth filtering mode will deliver results between the Mild and Aggressive approaches. It is also possible to disable the depth filtering mode, this is not advised.

**Figure 19: Dense point cloud**

To build the mesh choose the *Build Mesh* command from the *Workflow* menu. Following parameters can be set:

### Surface Type

**Arbitrary** surface type can be used for modelling of any kind of objects such as for example statues and buildings. **Height field** on the other hand requires less memory and is optimized for modelling of terrains or base reliefs. Height field is recommended for aerial photography

### Source Data

**Sparse Cloud** can be used for a fast build which is based solely on the sparse point cloud. For high quality results use **Dense Cloud** as source data. This will also result in longer processing Time.

**Figure 20: Comparison mesh source data sparse cloud and dense cloud**

#### Polygon Count

The polygon count specifies the maximum number of polygons in the final mesh. **High, Medium** or **Low** values are calculated based on the number of points based in the dense point cloud. The ratio values are 1:5 (High), 1:15 (Medium) and 1:45 (low). You still can indicate the target number according to your choice.

#### Interpolation

To achieve an accurate reconstruction interpolation should be **Disabled**. With interpolation disabled, manual hole filling is required at the post processing step. With **Enabled** PhotoScan will interpolate some surface areas. This way some holes will be automatically covered. **Extrapolated** will generate a holeless model with extrapolated geometry.

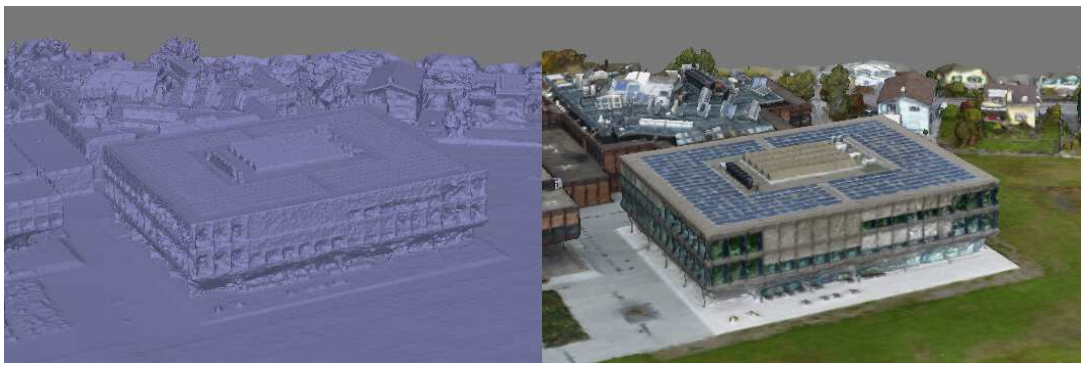

**Figure 21: Mesh uncolored and colored**

### 3.2.4.7 Edit Geometry and Texturing the Mesh

Sometimes it is necessary to edit the geometry before building the texture atlas or exporting the model. Unwanted faces can be removed. First indicate the faces using the selection tools from the toolbar. Then, to remove the selection use *Delete Selection* button on the *Toolbar*.

PhotoScan often creates 3D models with excessive geometry resolution. It is advised to decimate the mesh before exporting. To decimate the mesh select *Decimate Mesh…* from the *Tools* menu. For PDF export it is recommended to downsize the number of faces to about 200’000.

If you want to texture the mesh select *Build Texture* command from the *Workflow* menu. Different parameters can be set:

### Mapping Mode

This determines how the object texture will be packed in the texture atlas. **Generic** mapping mode is the default. It allows to parameterize texture atlas for arbitrary geometry. **Adaptive orthophoto** will split the object surface into flat part and vertical regions. The flat part is textured use in the orthographic projection, while vertical regions are textured separately to maintain accurate texture representation. To allow the whole surface to be textured in the orthographic projection use **Orthophoto** mapping mode. **Single photo** mapping mode will use a single photo to generate the texture. Which photo shall be used can be selected from *Texture from* list. **Keep UV**, this mode can be used to rebuild texture atlas using different resolution.

### Blending Mode

**Mosaic** blending mode first bends the low frequency components. High frequency components (creates the picture details) are taken from one picture. **Average** uses the weighted average of all the pixel values from each picture. **Max Intensity** uses the corresponding pixel of picture with the maximum intensity. **Min Intensity** uses the corresponding pixel of picture with the minimum intensity. It is possible to **Disable** the blending mode. In this case the colour value for the pixel is chosen like the one for the high frequency component in mosaic mode.

### Texture Size / Count

Texture size / count determines the width and height of the texture atlas (in pixels) and specifies the number of files for the texture export. Exporting the texturing to several files archives a greater resolution of the final model texture.

### Enable Color Correction

The colour correction process can be useful for data sets with extreme brightness variation. It takes quite a long processing time. Only use this feature for data sets that have proved to present results in poor quality.

### Enable Hole Filling

This parameter helps to avoid a salt-and-pepper effect in case of a complicated surface with numerous tiny parts of shading. It is recommend to only disable this function for very specific tasks.

**Figure 22: Textured model of the research centre**

### 3.2.5 Flowchart

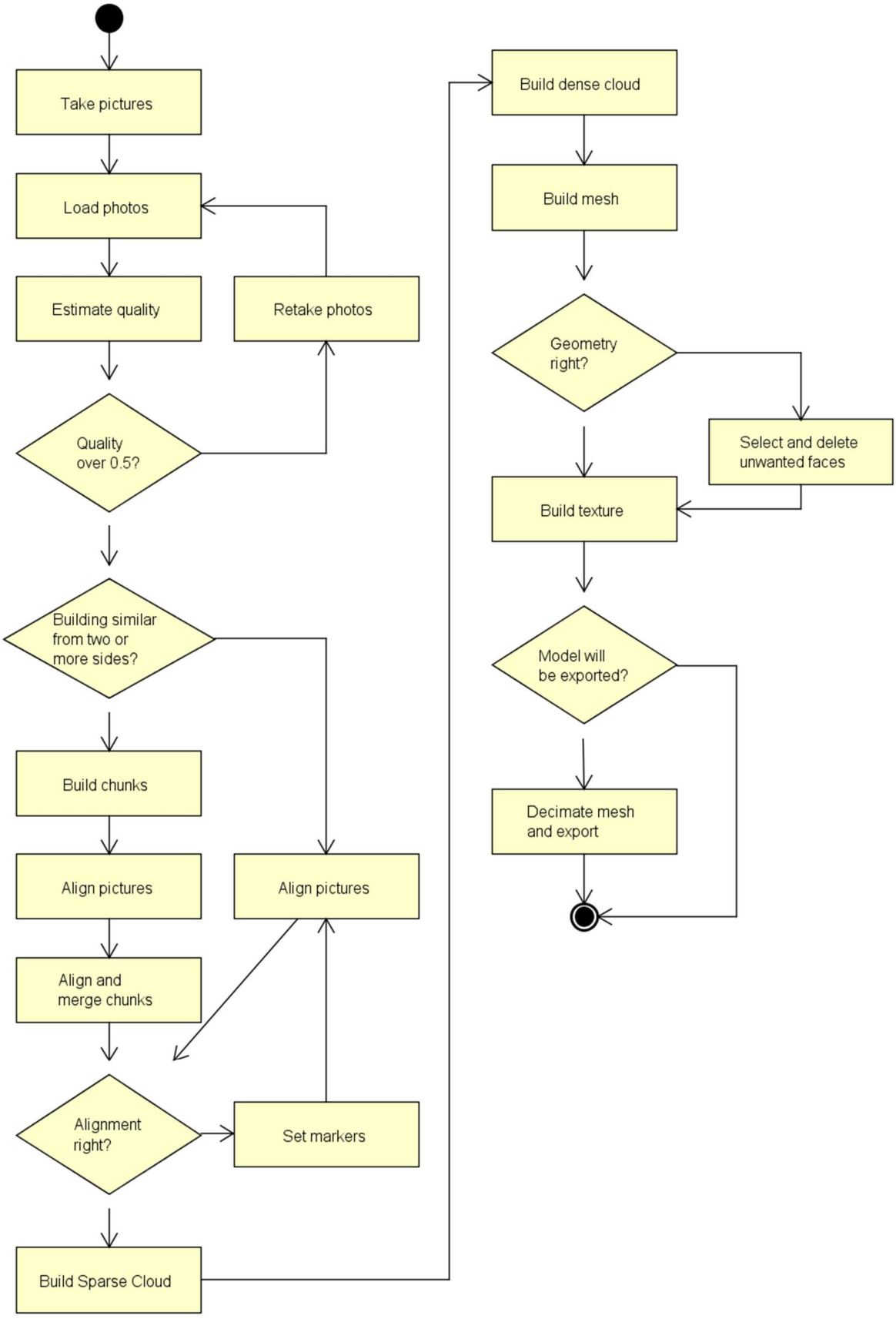


**Figure 23: Flowchart of the workflow with PhotoScan**

### 3.2.6 Pros and Cons

#### 3.2.6.1 Advantages

- Very realistic
- Faster to create than OSM with texturing
- Able to depict any object

#### 3.2.6.2 Downsides

- Tools and equipment costs
- No roofing
- Weather-dependent

### 3.2.7 Problems

If the building looks similar from different angles, the software seems to have trouble to distinguish one side from another and cannot align the pictures properly. To prevent this it is possible to manually set markers. These markers must be set as exactly as possible.

Another problem are the reflections of windows. One approach is to await perfect weather conditions. Another approach is to use a polarisation filter on the camera. The effectiveness of this approach has not been tested.

## 3.3 Workflow of Creating Textured 3D Meshes Using Pix4D and a Drone (Approach 2b)

### 3.3.1 Introduction

As mentioned above photogrammetry of buildings is limited to the front. To capture the roof you will even have to learn to fly or find a drone to do that for you. Together with the Pix4D software it is a quick and easy way to build a 3D model of buildings.

### 3.3.2 Requirements

For capturing a three-dimensional object with a drone and Pix4D you will need a drone that is compatible with Pix4Dcapture[12]. You will also need a smartphone.

The weather also has a great influence on the outcome. Rain may create reflections on flat surfaces. Sunny weather may cause reflections in windows. Optimal weather conditions would be light overcast.

### 3.3.3 Tools

#### 3.3.3.1 Pix4Dmapper

*Pix4Dmapper automatically converts images taken by drone, by hand, or by plane and delivers highly precise, georeferenced 2D maps and 3D models. They're customizable, timely, and complement a wide range of applications and software.*[13]

#### 3.3.3.2 Pix4Dcapture

*Pix4Dcapture turns your consumer drone into a professional mapping tool. A free companion of Pix4D software, Pix4Dcapture allows you to create flight plans for capturing image data.*[14]

#### 3.3.3.3 Drone DJI Phantom 4

*The DJI Phantom 4 is the smartest flying camera DJI has ever created. Able to fly intelligently with a tap, automatically create seamless tracking shots, fly intelligently over or around obstacles and much more.*[15]

### 3.3.4 Workflow

For capturing and generating 3D models with Pix4D and a drone lots of YouTube tutorials exists. Therefore this will be only a brief walkthrough with focus on explanations. You will also be led through the whole process by the applications.

[12] https://pix4d.com/product/pix4dcapture/#compatible-drones
[13] https://pix4d.com/product/pix4dmapper-pro/
[14] https://pix4d.com/product/pix4dcapture/
[15] https://store.dji.com/product/phantom-4

#### 3.3.4.1 Planning a Mission

First of all, you need to plan your flight missions in the Pix4Dcapture app. Select the double grid mission and position the grid. Make the grid a bit bigger than the object you want to capture, because the drone takes photos in a 70° angle. Pay attention to the altitude: if you set it to low, the drone will crash into trees or buildings.

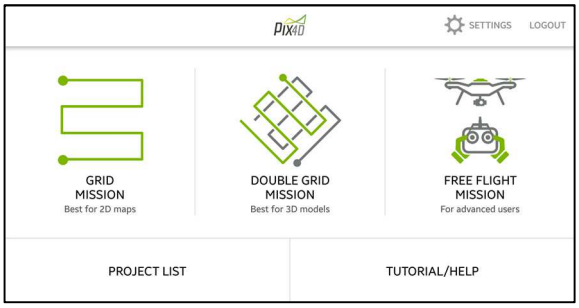


**Figure 24: Main window Pix4Dcapture**

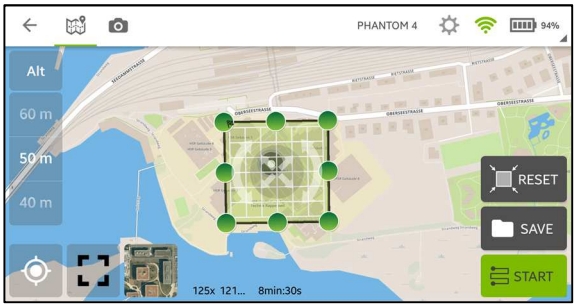


**Figure 25: Planning a mission**

The preset of the settings is quite well for getting usable photos. If you want to get even higher quality change the overlap value to 90%, but this will at least double the flight time.

Pix4Dcapture has an integratet project manager, where you can plan whole projekts with multiple missions. A benefit is the overview over what you have already captured.

#### 3.3.4.2 Take Pictures

Important: first clarify the legal situation with regard to privacy and data protection before flying with a drone.[16]

Be careful, especially if it is the first time you fly a drone. The drone is not faultless. Make sure you know how to abort a mission and how to land the drone manually before you start the mission. If you are unsure, make a test mission on a big meadow.

If you are happy with the mission and feel comfortable with the drone and the law, press start and you will be guided till the drone is finished with the mission.

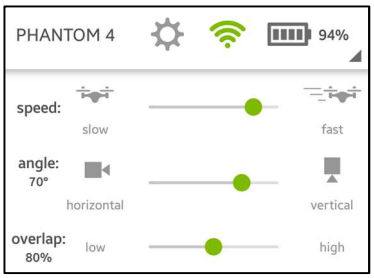


**Figure 26: Settings Pix4DCapture**

[16] More information can be found on the webpage of BAZL, see references

#### 3.3.4.3 Create Pix4Dmapper Project

After taking pictures, you will need to import them into Pix4Dmapper. To do that copy the pictures to the computer and create a new project in Pix4Dmapper. Give it a reasonable name and press the *Next*-button.

#### 3.3.4.4 Import Pictures into the Project

Now select the folder containing your pictures or add them individually. If you want to mix different flight missions, you should make sure that the daytime and thus the brightness of the object from the flight missions are not too different. Else you will have displacements or colour spots. Select all pictures from one mission, or if you made multiple flight missions from the same object then from all these missions, and change their group to a reasonable name. There is a simple way how to do that, just click the first image of a mission and then shift click the last one of this mission. Right click the column *Group* in any row and press *Edit Groups in Selected Rows*. Now type the name in and click somewhere else to submit. If you are finished press *Next* twice.

#### 3.3.4.5 Start Processing

You should now be able to choose the processing options template; take 3D models or 3D models – rapid/low res. Check the *Start Processing Now*-checkbox or deselect it if you want to change the settings more specifically. Press *Finish* to continue. In the menu *Process* you will find the processing options. If the process is not already running you can start it after changing the settings by pressing the *Start*-button in the processing bar at the bottom of the window.

##### Initial Processing

During the initial process, the images are matched, unsuitable images are rejected and their exact position is triangulated using key points. This key points are also called tie points. The image scale setting has a direct influence to the accuracy of the result.

##### Point Cloud

After the camera positions are fixed, the program will search for pixels that appear in multiple images. The result is a cloud of points that gives you a quite detailed view of the resulting mesh. As before you can modify the quality of the output, but the higher the quality, the longer it takes. A higher point density will make your 3D model rougher. If you have a building with flat faces use a lower setting, but if the surface is rough use a higher setting. It is not suggested to change the **Minimum Number of Matches** except when you have a high overlay of pictures.

##### Mesh

Pix4Dmapper will use the points from the point cloud to generate the 3D textured mesh. You can choose in which output type the mesh should be saved. To choose the quality of the texture is similar to choosing the quality of a picture, if you need to go close use a high quality, but do not set it higher as needed. Unreal Engine for example can handle only textures with a maximum of 8192x8192 pixels.

#### 3.3.4.6 Process Finished

Pix4Dmapper will run the whole process without any break and at the end of each part it will display a detailed report. If the program is finished the output files are saved in a folder next to the project file.

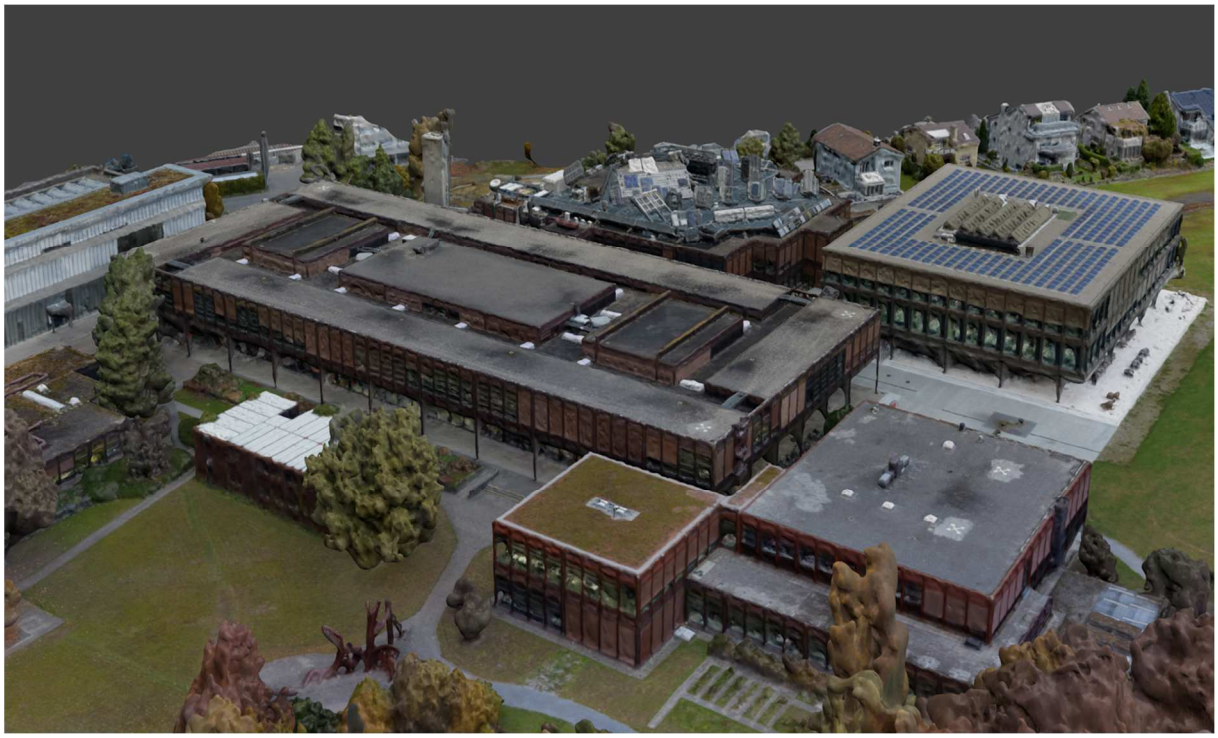

**Figure 27: Model generated by Pix4D**

### 3.3.5 Flowchart

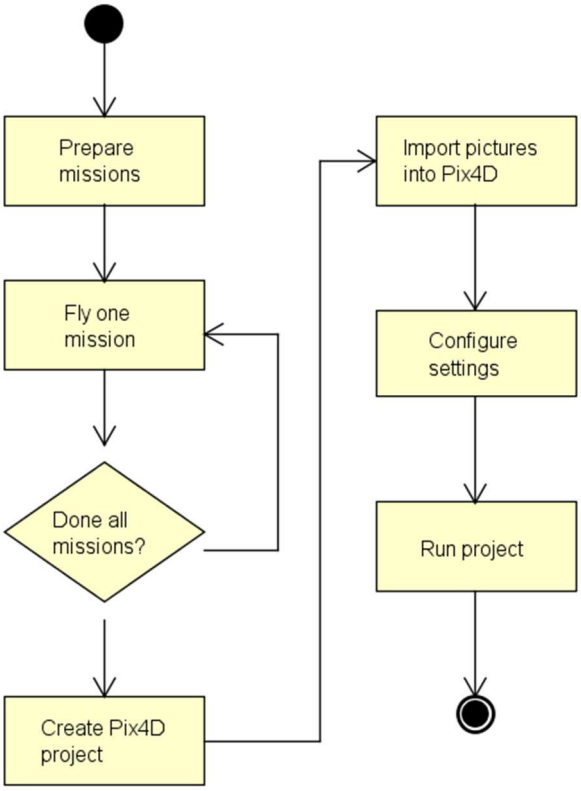


**Figure 28: Flowchart of the workflow with drone and Pix4D**

### 3.3.6 Pros and Cons

#### 3.3.6.1 Advantages

- Very realistic
- Drone flies autonomous
- Fast capturing of big areas
- Simple workflow
- Area can easily be extended

#### 3.3.6.2 Downsides

- Cost intensive
- Good drone needed
- Difficult to map areas covered by overhanging parts

### 3.3.7 Problems

If you want to capture something like a big sculpture, where you need to fly multiple times in different perspectives, you will have lot difficulties. The changing sun position and thereby changing light conditions cannot be processed correctly by Pix4Dmapper.

Some drones will have problems if you fly to fast. The pictures are then blurred. This can be avoided by setting the drone to fly slowly.

## 3.4 Comparing Agisoft and Pix4D

Both Agisoft PhotoScan and Pix4D are incredible powerful photogrammetry tools. Each software is able to build a 3D model based on multiple pictures of an object. Both models are highly realistic, show many details and are suitable as game environment.

Agisoft provides the user with more control over the process. The parameters for each step of the process can individually be set to reach best possible results. This also implies a certain understanding of the process by the user. Generally Agisoft PhotoScan is faster at processing the data than Pix4D[17].

Pix4D is specialized to be used in combination with a drone. It offers an app designed to plan and fly missions for the drone. Next to the app you also get access to cloud storage. The software is easy and intuitive to use and requires practically no induction. Pricewise Pix4D is higher, but offers some useful additions.

**Figure 29: Model generated with Pix4D**

**Figure 30: Model generated with Agisoft PhotoScan**

[17] https://eprints.hsr.ch/439/

# 4 Conclusion: The Results and What We Have Learned

If you want to base your project on open source data and software using OSM and OSM2World will give you fast and good results. It is only the texturing which is time consuming. If one is used to working with blender then even the texturing will not take long. It still remains a difficult task to create a realistic texture. The results are usable as enviroment for a game such as a racing game.

A community of volunteers have built the massive world of OSM. The combination of passionate programmers and an easy extendable open source programm such as OSM2World can lead to surprising results. This means that with time it could be be possible that the basic skills of OSM2World can progress so far as to already include texturing. By extending OSM2World making it possible to depict tables with chairs we have proven that OSM2World can easily be extended.

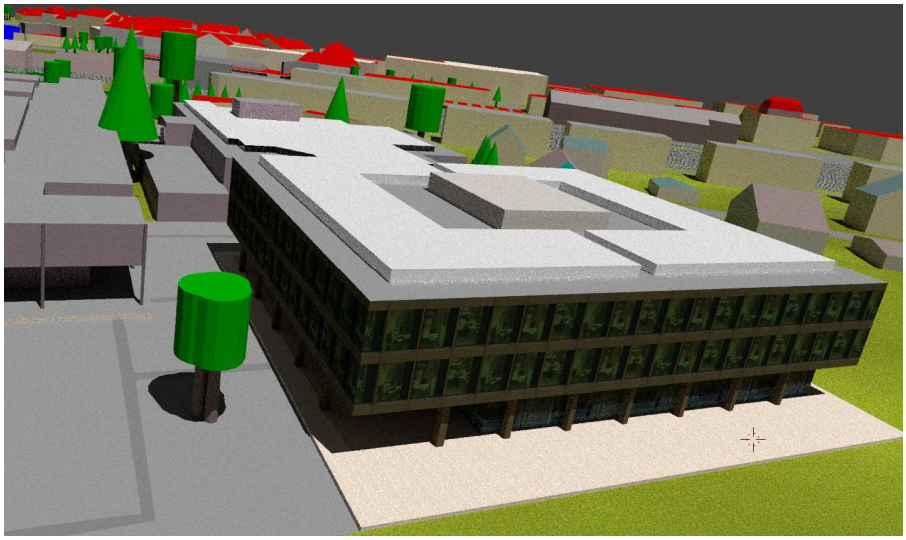

**Figure 31: Model generated from OSM and textured with Blender**

The approach of using a camera with Agisoft PhotoScan is not the best solution. With the roof an important part of the building cannot be modelled. Next to that it is almost impossible to get a good model of a building with as many windows as the research centre. It became clear that taking pictures manually is not as easy as expected.

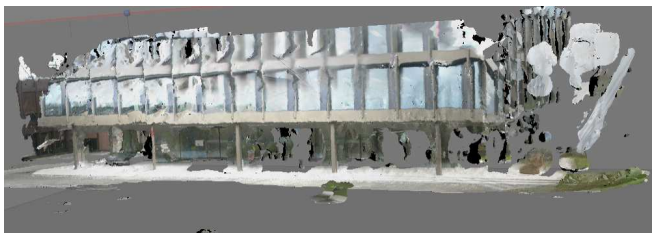

**Figure 32: Front side of the research centre; photos taken manually**

The drone and Pix4Dmapper deliver astonishing results, very detailed and realistic. The only problems lie with overhanging building parts as the drone takes all the pictures from above. The model would be great for a game played from an aerial overview, like a drone racing game.

**Figure 33: Model of the research centre built with Pix4D**

It does not really depend which photogrammetry tool you choose, both Agisoft PhotoScan and Pix4Dmapper are very powerful. It is advisable to take the pictures with a drone. The opportunity of free flights allows you to also take pictures from the front of the building. There is also the possibility to add manually taken pictures.

Depending on the use of the model, resources, area and requirements towards the model, both the model generated from OSM data and the model built from drone photos are suitable. We want to program a game in highly realistic game environment. Drone and software license are available to us. Next to that Agisoft PhotoScan is a bit faster than Pix4Dmapper and lets us have more control over the model building process. Based on these arguments we decided to use a model based on drone pictures and built with Agisoft PhotoScan.

# 5 Outlook: What Was Not Covered and What Will Happen Next

The topic of generating 3D models is wide spread. As a consequence this thesis focused on only three approaches. Another approach would be using a laser scanner. LIDAR creates a dense point cloud using laser rays to measure the distance. There may also be more new technologies depict the reality as a 3D model. In both Pix4D and Agisoft PhotoScan there should be the possibility to draw the base geometry of an object and then map the texture from the photos to this object. This approach was not evaluated.

A promising approach would also be combining photos taken by drone and photos taken manually on ground. The pictures taken manually on ground would cover the parts the drone cannot. This approach could be very convenient with overhanging parts like balconies.

Also an interesting section of photogrammetry would be indoor. This would allow new uses for the model such as a “walk around campus map”. Google Tango could prove to be helpful here. Tango is a platform that uses computer to enable mobile devices to detect their position relative to the world around them. Special Hardware is needed to run Tango. For more information look up the Tango Webpage.

An Unreal Engine developer has published a plugin for OSM. It lets you drag and drop OSM data into the content browser of Unreal Engine and save it as an asset. As this has only recently been published there was not enough time to try it.

Towards the end of our project Pix4D released a new version promising better recognition of flat surfaces. There was no time left to compare these two versions in this term project.

The results of this project are the basis for a follow-up project called "Interactive Multiplayer HSR-Game with Motion”. Two 3D models are available now as a starting point, ready to be integrated into a game engine.

**Figure 34: Possible token of the game**

It was decided to use a model created with the drone and Pix4D. To have enough space to play another model will be created using the drone. The model should cover at least the whole campus. Additionally parts of the lake and field behind the research centre would be desirable. The game will then be played in the airspace above the HSR campus. An idea would be a racing game with drones.

# 6 Personal Reports

## 6.1 Report Sophie Somerton

This was a very interesting project, not your typical everyday IT-project. Software evaluation is often underrated. It is important to choose the right software to get the best results. It was great fun to use a powerful program such as Agisoft PhotoScan or play around with an impressing drone like the phantom 4. The results were astonishing.

I was surprised of the amount of interest over disciplines showed towards this thesis. It was interdisciplinary from the beginning as the main stake holder was from the electronical department. Towards the end the department of landscape architects hired me to build a model of a specific building.

There was much to be learned throughout this project. For me the most important concerns project management. Our time management failed due to not having a specific target from the beginning. This meant there were major delays in organising all the needed equipment and software.

I am really looking forward to the follow-up project. It is great to know that we are going to have a great game environment for our game. We intend not to repeat all management mistakes made throughout this project.

## 6.2 Report Joel Hochreutener

It was an interesting and challenging project. I was able to learn a lot and we were strongly supported by different people. I was really happy when we got the opportunity to fly with a drone. Thanks to the powerful computer, we were able to process even big data sets.

At first, I could not imagine all the possibilities having OSM and a passion for details, but now I find it a pity that we already have come to an end. I have become very fond of Blender. All the more I look forward to working with a game engine and the 6DOF Motion-Simulators.

Unfortunately our time management was not very successful, which we definitely need to change when planning our next project.

I would like to thank Prof. Stefan Keller and the ICOM for making this possible.

# 7 References

## 7.1 Glossary

| | |
|---|---|
| BIM | Building Information Modelling |
| DOF | Degrees of Freedom |
| GIS | Geographical Information Systems |
| HSR | University of Applied Science Rapperswil |
| OSM | OpenStreetMap |
| Photogrammetry | Science of making measurements from photographs |
| Mesh | Collection of vertices, edges and faces to define the shape of a polyhedral object in 3D computer graphics and slid modelling |
| Point Cloud | Set of data points in a coordinate system |
| Tie Point | A point that has ground coordinates that are not known, but is visually recognizable in the overlap area between two or more images |
| Orthophoto | Geometrically corrected aerial photograph |
| LIDAR | Light Detection and Ranging, measures distance to a target by illuminating that target with a laser light |
| Tango | Technology platform based on motion tracking, area learning and depth perception |
| Ant build file | File that uses XML to describe the build process and its dependencies |
| UV mapping | Projecting a 2D image to a 3D model's surface |
| Texture atlas | Overall UV map, not just one object |

## 7.2 Table of figures

## 7.3 Resources

Agisoft PhotoScan user manual, 2016, http://www.agisoft.com/pdf/photoscan-pro_1_2_en.pdf

Bericht zivile Drohnen, BAZL, 10.2.2016, https://www.bazl.admin.ch/bazl/de/home/gutzuwissen/drohnen-und-flugmodelle.html#70_1444378848725__content_bazl_de_home_gutzuwissen_drohnen-und-flugmodelle_jcr_content_par_tabs

Chapters 6.2 and 7.2, term project “Photogrammetrie-Evaluation”, Bargen Danilo und Schmid Josua, 12.10.2015, https://eprints.hsr.ch/439/

Personal conversation with Marianne Deuber, Terradata, 10.10.2016, Sitzungsprotokolle.docx, page 5

# 8 Acknowledgements

We would like to thank following people for the support we received throughout this project:

- Prof. Stefan Keller, IFS
- Robert Hegner, ICOM
- Prof. Guido Schuster, ICOM
- Rainer Schaufelberger
- Marianne Deuber, Terradata

# Appendix

## A. Project Management

All project data, documents and pictures can be found on GitHub using following URL:

https://github.com/cjhox/VRmotion_HSR-Game

## B. List of Software

| Name | Version | License | Cost | URL |
|---|---|---|---|---|
| Agisoft PhotoScan | Prof. Version 1.2.6 | Proprietary | Professional Version 3'499 USD, Standard Version 179 USD.[18] | http://www.agisoft.com/ |
| Blender | 2.78a | GPL | Free | https://www.blender.org/ |
| OpenStreetMap | - | ODbL | Free | https://www.openstreetmap.org |
| OSM2World | 0.1.9 | LGPL | Free | http://osm2world.org/ |
| Pix4Dmapper | Prof. Version 3.0.17 | Proprietary | Subscription, yearly 3200 CHF, Perpetual 7900 CHF, educational License 4900 CHF | https://pix4d.com/ |
| Pix4Dcapture | 3.3.1 | - | Free | https://play.google.com/store/apps/details?id=com.pix4d.pix4dmapper&hl=en |

## C. Code of Table Class for OSM2World

```
private static final class Table extends NoOutlineNodeWorldObject
      implements RenderableToAllTargets {

   private final ConfMaterial defaultMaterial;

   public Table(MapNode node) {
      super(node);

      if (node.getTags().contains("leisure", "picnic_table")) {
         defaultMaterial = Materials.WOOD;
      } else {
         defaultMaterial = Materials.STEEL;
      }
   }

   @Override
```

[18] Educational licenses available to special conditions.

```
   public GroundState getGroundState() {
      return GroundState.ON;
   }

   @Override
   public void renderTo(Target<?> target) {

      int seats = parseInt(node.getTags(), 4, "seats");

      // All default values are bound to the hight value. This allows to chose any
table size.
      float height = parseHeight(node.getTags(), 0.75f);
      float width = parseWidth(node.getTags(), height * 1.2f);
      float length = parseLength(node.getTags(), ((seats + 1) / 2) * height /
1.25f);

      float seatHeight = height / 1.5f;

      /* determine material */

      Material material = null;

      //TODO parse color

      if (material == null) {
         material = Materials.getSurfaceMaterial(
               node.getTags().getValue("material"));
      }

      if (material == null) {
         material = Materials.getSurfaceMaterial(
               node.getTags().getValue("surface"), defaultMaterial);
      }

      /* calculate vectors and corners */

      double directionAngle = parseDirection(node.getTags(), PI);

      VectorXZ faceVector = VectorXZ.fromAngle(directionAngle);
      VectorXZ boardVector = faceVector.rightNormal();

      float poleCenterOffset = height / 15f;

      List<VectorXZ> cornerOffsets = new ArrayList<VectorXZ>(4);
      cornerOffsets.add(faceVector.mult(+length / 2 -
poleCenterOffset).add(boardVector.mult(+width / 2 - poleCenterOffset)));
      cornerOffsets.add(faceVector.mult(+length / 2 -
poleCenterOffset).add(boardVector.mult(-width / 2 + poleCenterOffset)));
      cornerOffsets.add(faceVector.mult(-length / 2 +
poleCenterOffset).add(boardVector.mult(+width / 2 - poleCenterOffset)));
      cornerOffsets.add(faceVector.mult(-length / 2 +
poleCenterOffset).add(boardVector.mult(-width / 2 + poleCenterOffset)));

      /* draw poles */

      float poleThickness = poleCenterOffset * 1.6f;

      for (VectorXZ cornerOffset : cornerOffsets) {
         VectorXZ polePos = node.getPos().add(cornerOffset);
         target.drawBox(material, polePos.xyz(getBase().y + 0.001), faceVector,
height, poleThickness, poleThickness);
      }

      /* draw table */

      target.drawBox(material, getBase().addY(height * 14f / 15f),
            faceVector, height / 15f, width, length);

      /* draw seats */
```

```
      int leftSeats = seats / 2;
      int rightSeats = (seats + 1) / 2;

      renderSeatSide(target, material, boardVector.mult(+width / 2 + seatHeight /
2.5f), length, leftSeats, seatHeight);
      renderSeatSide(target, material, boardVector.mult(-width / 2 - seatHeight /
2.5f), length, rightSeats, seatHeight);
   }

   private void renderSeatSide(Target<?> target, Material material, VectorXZ
rowPos, float length, int seats, float seatHeight) {
      VectorXZ boardVector = rowPos.rightNormal();
      VectorXZ faceVector = rowPos.normalize();

      float seatWidth = seatHeight / 1.25f;
      float seatLength = seatHeight / 1.25f;
      VectorXYZ seatBase = getBase().add(rowPos).addY(seatHeight * 0.94f);
      VectorXYZ backrestBase = getBase().add(rowPos).add(faceVector.mult(seatLength
* 0.45f)).addY(seatHeight);

      for (int i = 0; i < seats; i++) {
         float seatBoardPos = length / seats * ((seats - 1) / 2.0f - i);

         VectorXZ seatPos = rowPos.add(boardVector.mult(seatBoardPos));

         List<VectorXZ> cornerOffsets = new ArrayList<VectorXZ>(4);
         cornerOffsets.add(seatPos.add(boardVector.mult(+seatWidth *
0.45f).add(faceVector.mult(+seatLength * 0.45f))));
         cornerOffsets.add(seatPos.add(boardVector.mult(+seatWidth *
0.45f).add(faceVector.mult(-seatLength * 0.45f))));
         cornerOffsets.add(seatPos.add(boardVector.mult(-seatWidth *
0.45f).add(faceVector.mult(-seatLength * 0.45f))));
         cornerOffsets.add(seatPos.add(boardVector.mult(-seatWidth *
0.45f).add(faceVector.mult(+seatLength * 0.45f))));

         /* draw poles */

         for (VectorXZ cornerOffset : cornerOffsets) {
            VectorXZ polePos = node.getPos().add(cornerOffset);
            target.drawBox(material, polePos.xyz(getBase().y + 0.001), faceVector,
seatHeight, seatWidth / 10f, seatLength / 10f);
         }

         /* draw seat */


         target.drawBox(material,
               seatBase.add(boardVector.mult(seatBoardPos)),
               faceVector, seatHeight * 0.06f, seatWidth, seatLength);
```

# D. Additional Documents

To this documentation there can be found following additional documents (all written in German):

- Project plan: Projektplan.docx
- Project structure: Projektstruktur.docx
- Scope of the term project: SA_Aufgabenstellung.docx
- Technical risks: Technische_Risiken.xlsx
- Minutes of meetings: Sitzungsprotokolle.docx
- Project management analysis: Auswertung.docx