Point Cloud Generation Based on Aerial Imagery

3D models of urban areas are becoming increasingly popular together with the number of applications they are used for. Examples of such are urban planning, potential flood analysis, solar potential estimation, development of smart cities alongside further real-time simulation purposes. Usually, city models require multiple harmonized data sources to be constructed. A simple building model would need building footprints with additional attributes, such as height information and a Digital Terrain Model (DTM) to generate a relief. Even though open-source data such as Open Street Map (OSM) exists, most of accurate data is not publicly available and has to be acquired from an administrative agency or a private company for a fee. Therefore, this procedure often proves to be expensive and sometimes, if the necessary data is not available, impossible to conduct.

The ability to generate 3D models based on singular data sources, such as images, would allow users to reconstruct areas in urban scenes with very little input information. This is advantageous since many regions around the world do not possess enough data to generate reliable models. The following project work aims to research and implement the first step in generating semantic 3D models from 2D imagery, namely by gaining insight on existing solutions that can generate artificial 3D point clouds. To generalize the problem, the sole input for the reconstruction is single-view (nadir) aerial imagery.

The project work can be separated into two general sections. The first one includes researching suitable methods can be utilized to generate point clouds, whereas the second encompasses implementing the chosen approach.

Research

At this point, it is important to distinguish between aerial and satellite imagery. A main aspect between both types is the altitude of the sensor, which then delivers images with varying ground sampling distances and therefore varying detail of the objects. Having said that, aerial imagery provides more detail, which is necessary for the task at hand. Additionally, aerial datasets have a larger overlap, which is an essential when conventional methods are utilized.

Three possible paths for point cloud generation were investigated based on existing solutions, and available datasets:

- Photogrammetric
- LSD-SLAM
- Deep Learning based

The dataset used in the implementation is DFC2018 [1]. It contains a Digital Surface Model and a Digital Elevation Model generated from multispectral LiDAR data, high resolution RGB imagery, and several urban land-use classes. COLMAP [2], [3], [4] was used for the photogrammetric reconstruction. However, the scene could not be reconstructed through Structure from Motion (SfM) because the image

overlap is low or non-existent. On the bright side, photogrammetric reconstruction is a well-established method and is not of big interest for this work, apart from its possible use as a baseline for comparison.

A very interesting method to be directly tested for suitability is LSD-SLAM [5], since it is conceived to compute point clouds out of multi-view imagery around a specific object, whereas the images are taken in a rapid succession with a small Euclidean distance between them. Comparing the scenario in the project with the SLAM approach, aerial images here have a much larger baseline and capture objects further away from the sensor. Additionally, the scene is much larger than what LSD-SLAM is designed for. Unfortunately, the software could not be tested successfully because it proved extremely difficult to reproduce its execution environment. Initially, Virtual Box was set up with an Ubuntu image as a test environment. The ROS (Robot Operating System) libraries and the Ubuntu version required to run the algorithm, however, were outdated which obstructed running them in a virtual machine due to unmet dependencies in the software. To combat this, it was attempted to run LSD-SLAM in a Docker container, which has been created in the past and where the mentioned dependencies are solved. The tested containers are [6], [7], [8], [9]. Executing the containers was unsuccessful in every case.

After the unsuccessful use of LSD-SLAM, the focus was shifted to the Deep Learning based approaches. Conceptually, it is possible to exploit the image overlap in a dataset and use it for training or take advantage of the motion parallax during the movement of the sensor. However, no such reliable implementations were found. Therefore, papers introducing deep learning architectures that exhibit a pixel-based approach and deliver depth images were chosen and examined [10], [11], [12], [13], [14]. The following aspects were compared in order to choose a reliable solution out of the found ones:

- Availability of training data
- Publicly available code
- Accuracy
- Up-to-dateness
- Output (Multi-Task Learning MTL)
- Easy to implement due to the given time frame.

Ultimately, [10] was chosen because it is covers up all aforementioned points. The model is trained on the DFC2018 dataset, which is not georeferenced. The corresponding repository can be found in [15]. It consists of two parts: a convolutional encoder-decoder neural network, where a pre-trained DenseNet121 is used as a main encoder, and an additional autoencoder that removes noise from the output depth image. Additionally, the first part of the architecture can deliver pixel-based segmentation masks and normal vectors for the input images. A docker container is used here as well to isolate and make use of the ready-to-use Tensorflow environment, which can also be challenging to set up. Training time with Transfer Learning for ten images was approximately 150 minutes on a NVIDIA GeForce GTX 880M. The pre-trained weights are also to be found in the model repository. Testing the neural network in Windows can be done as follows:

Install Docker and Windows Subsystem for Linux

Pull a Docker image that is suitable for running python scripts. In this case, an image with Python and Tensorflow 2.1 with CUDA is required

- `docker pull sha256:1010e051dde4a9b62532a80f4a9a619013eafc78491542d5ef5da796cc2697ae`

Run a container from the downloaded image. In this case, we run the container with the root user.

- `docker run -it tensorflow/tensorflow:2.1.0-gpu-py3`

Create a directory for the model:

- `mkdir model`

Re-synchronize the package index files from their sources

- `apt-get update`

Download git repository, compress to .tar and copy to docker container through the command line. Copy command structure: docker cp local path, container id: container path

- `docker cp D:\BigData\DSMNet-main.tar 7259c6f58174f8fca4bf955c0d43ab7d00165eacf083e3baed7f057747651ec0:model`

Extract copied file

- `(sudo apt-get install xz-utils)`
- `tar -xf DSMNet-main.tar`

Install necessary python libraries

- `pip install opencv-python-headless`
- `pip install pillow`
- `pip install scikit-image`

Test network

- `python test_dsm.py DFC2018 True`

Copy model output locally for further processing:

- `docker cp 1ca4dbc7617340d234276284d127a9b03753ce7fd4d243a0e0fce19e53dbd670:test/DSMNet/output D:\output`

Workflow 1: Executing DSMNet through Docker

Results

The output can be seen in Figure 1 and is a depth image with similar properties as the Digital Surface Model, since height maps are generally a result of DSM-DTM. However, it exhibits a gaussian smoothened surface with fewer details. For clarity, these differences will be shown when evaluating the accuracy of the model. The DSM, which can be considered as ground truth, is derived from a LiDAR point cloud, and has a 0.5m GSD.

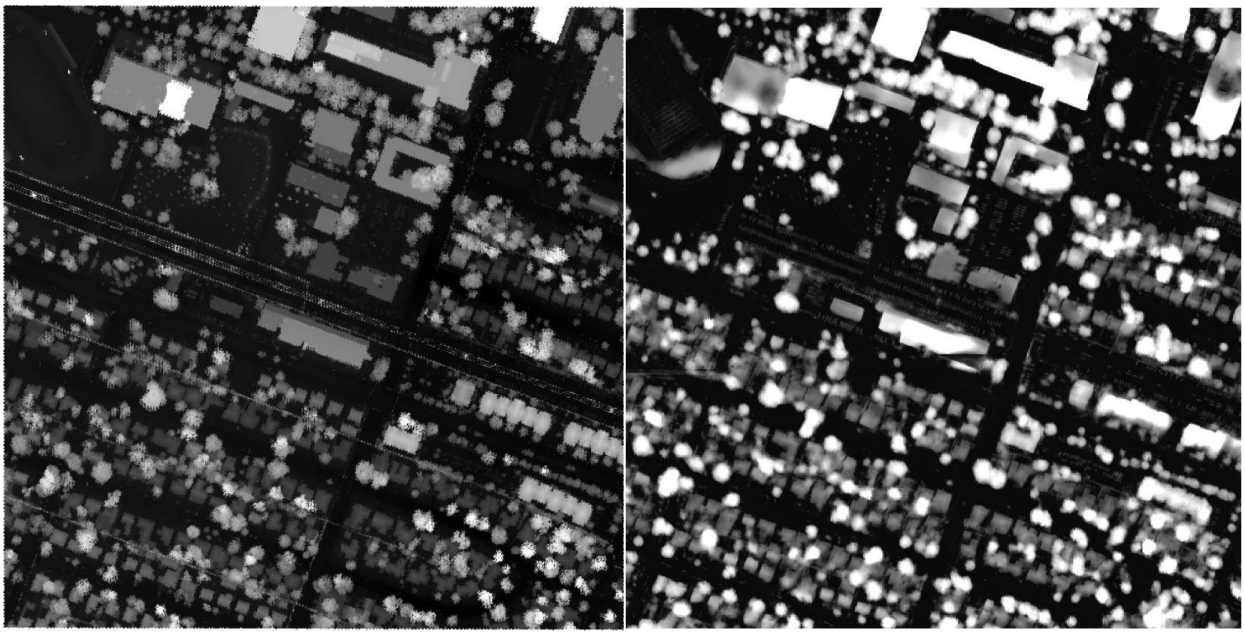

Figure 1: DSM obtained from multispectral LiDAR (left) and depth image from model inference (right)

Once the depth image has been predicted, the point cloud be generated as well. For this, a simple Python script that transforms pixel information to a vector was written. Additionally, two separate workflows that can read and write standardized point cloud formats, namely .ply and .las, are included. The code can be found in [16].

Now that the point cloud has finally been generated, it is visualized and evaluated against the DSM. For this, the ground truth is also transformed to a point cloud. Visualization can be performed in different ways: inside Python with the Open3D library, by using ArcGIS, or a point cloud software such as CloudCompare. The latter is implemented since it is also capable of manipulating and evaluating point clouds.

Due to scaling issues, the point clouds are displaced from one another and need to be further transformed. The DSM derived point cloud is taken as a baseline. Figures 2 and 3 illustrate the results and give first insights on their differences.

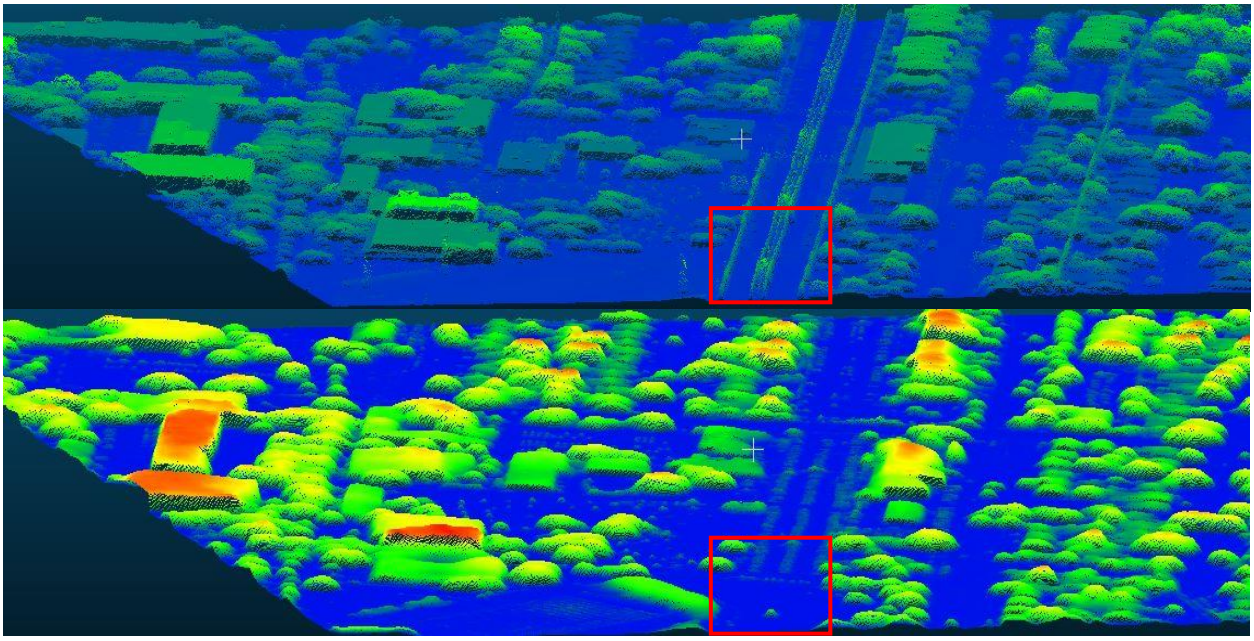

Figure 2: Point Clouds derived from DSM (top) and from the Deep Learning model (bottom)

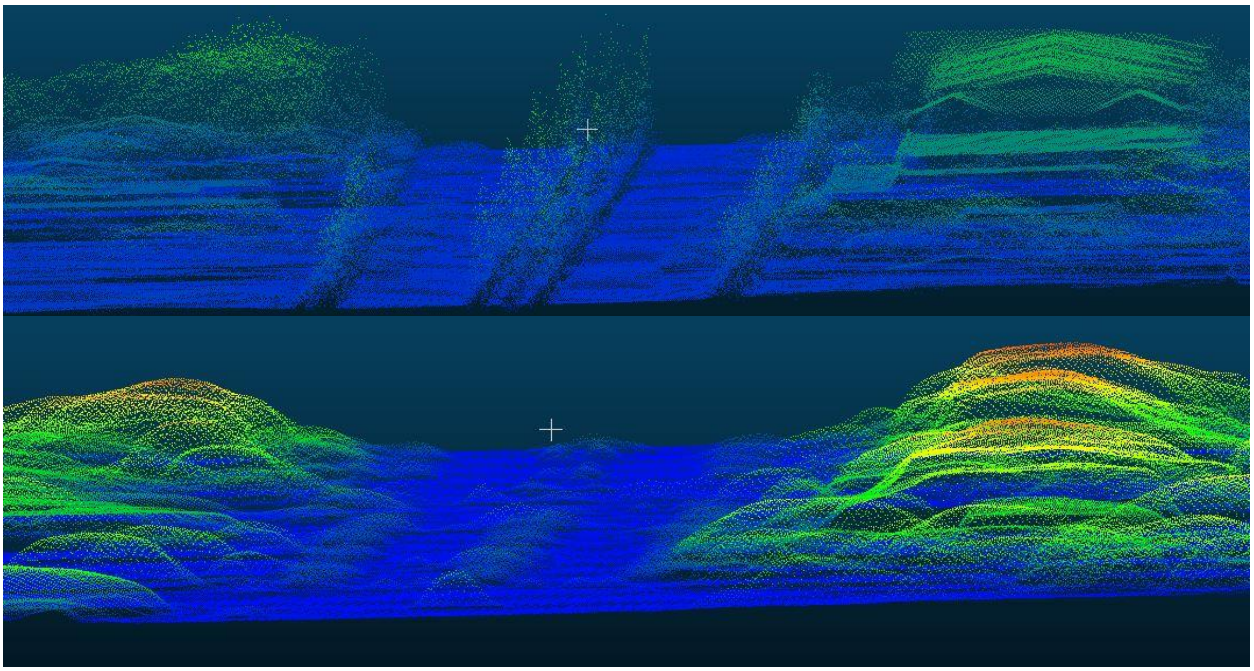

Figure 3: Power lines are visible in the DSM point cloud (top) and not at all captured by the Deep Learning model (bottom)

As a first impression, the separation of the objects is clear and the relative heights in the DSMNet point cloud are correctly generated. The authors in [10] achieved this by adding gaussian smoothing to deal with overlapping areas. This affects, however, the sharpness of building edges, which is problematic when creating semantic city models. Trees, having a rounder surface, which responds better to gaussian kernels, have been reconstructed sufficiently well. Finer structures, such as the overhead power lines are not captured by the model. This is an expected results because the GSD of the input images is too high. Photogrammetrically derived point clouds would also fail to reconstruct such detailed features, leaving only LiDAR as a viable option.

Computing the cloud-to-cloud distances between the outputs results in the following statistical values and corresponding figure:

- Mean distance = 1.37359 m
- Standard deviation = 1.20883 m

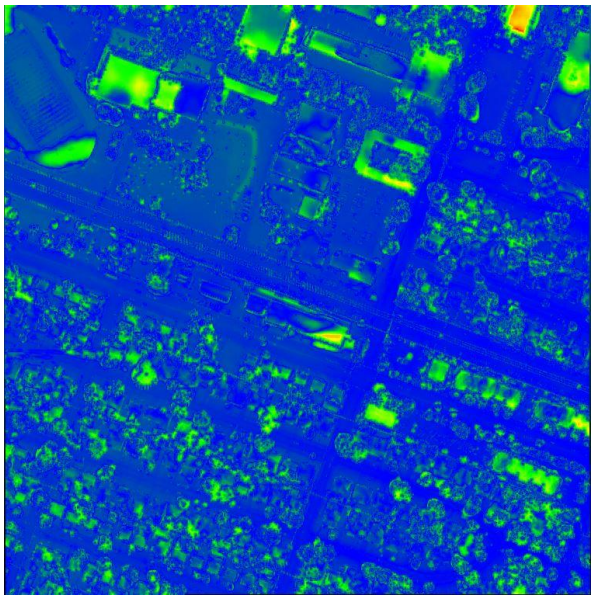

Figure 4: Cloud-to-cloud distances between the model result and ground truth

Near-correct correspondance of the point cloud results in a blue color, deviations from the DSM point cloud are depicted in a green-to-red scale. From what is visible in the point clouds, the height difference between the objects is spread widely, meaning that a lot of point possess no differences and others – large ones. This is not deducible from the calculated mean distance. For example, the ground points in the newly generated point cloud mostly have heights similar to the ground truth. However, higher structures, especially buildings, often have significantly larger deviations than the mean. It can be argued that DSMNet tends to generate points with larger deviations from the ground truth, the higher the reconstructed objects are. On the other hand, the small house blocks in the lower half of the image possess a good correspondance to the ground truth.

Difficulties

Several aspects during this project work proved to be challenging. First of all, implementing LSD-SLAM was unsuccessful, even though different methods were attempted to reproduce the software environment. Additionally, working with Docker was challenging. The software has a steep learning curve with documentation that is badly structured and not very understandable. However, once one learns how to deploy and use containers, it is a very powerful and useful tool. For this, gaining knowledge on Docker was a key aspect in this project. Using the Open3D Python library was also challenging for similar reasons. Initially, it was planned to utilize it for the point cloud generation but that attempt proved unsuccessful. On the other hand, it is quite easy to write point cloud formats with the library. ArcGIS Pro was the first software used for matching and analyzing the point clouds, but it does not possess the necessary functionalities for the task at hand.

Conclusion

Although it is a novel and robust approach, reconstructing 3D areas with neural networks is still not precise and accurate enough for semantic city modelling, especially where roof structures are needed for solar potential estimation. Edges of buildings are not precise enough to capture areas accurately and the smoothing results in curvatures where areas are supposed to be flat. The extracted height map can be quite beneficial in cases, where only cadaster plots and no height information are available. Furthermore, these models would be useful in large-scale scenarios where precision is not as important, such as topography modelling.

References

[1] 2018 IEEE GRSS Data Fusion Challenge – Fusion of Multispectral LiDAR and Hyperspectral Data. https://hyperspectral.ee.uh.edu/?page_id=1075

[2] COLMAP – SfM and MVS. https://demuc.de/colmap/

[3] Schönberger J., Frahm J., Structure-from-Motion Revisited. DOI: 10.1109/CVPR.2016.445

[4] Schönberger, J.L., Zheng, E., Frahm, JM., Pollefeys, M. (2016). Pixelwise View Selection for Unstructured Multi-View Stereo. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9907. Springer, Cham. https://doi.org/10.1007/978-3-319-46487-9_31

[5] Engel, J., Schöps, T., Cremers, D. (2014). LSD-SLAM: Large-Scale Direct Monocular SLAM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8690. Springer, Cham. https://doi.org/10.1007/978-3-319-10605-2_54

[6] SLAM Environment Docker https://hub.docker.com/r/kosuke49/ros_slam

[7] A Dockerfile for LSD-SLAM https://github.com/ozakiryota/docker_lsdslam

[8] https://github.com/IshitaTakeshi/lsd_slam_noros

[9] https://github.com/tum-vision/lsd_slam/issues/342

[10] Elhousni M., Zhang Z., Huang X. Height Prediction and Refinement from Aerial Images with Semantic and Geometric Guidance. https://arxiv.org/pdf/2011.10697.pdf

[11] Xu Y. et al., Advanced Multi-Sensor Optical Remote Sensing for Urban Land Use and Land Cover Classification: Outcome of the 2018 IEEE GRSS Data Fusion Contest. https://ieeexplore.ieee.org/document/8727489

[12] Carvalho M., Saux B., Trouve P., Multi-Task Learning of Height and Semantics from Aerial Images. https://arxiv.org/pdf/1911.07543.pdf

[13] Liu C. et al., IM2ELEVATION: Building Height Estimation from Single-View Aerial Imagery. https://doi.org/10.3390/rs12172719

[14] Karatsiolis S., Kamilaris A., Cole I., IMG2nDSM: Height Estimation from Single Airborne RGB Images with Deep Learning. https://doi.org/10.3390/rs13122417

[15] melhousni/ DSMNet. https://github.com/melhousni/DSMNet

[16] hristiyand/ BigDataProject. https://github.com/hristiyand/BigDataProject