Master Thesis, August 2018

Image Segmentation using Convolutional Neural Networks for Change Detection of Landcover

Intelligence Augmentation using Open Data and Deep Learning

Author
Martin Boos

Supervisor
Prof. Stefan Keller

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

MSE | MASTER OF SCIENCE IN ENGINEERING

This page is intentionally left blank.

Abstract

Periodically, cadastral survey data has to be updated. Generally spoken, this means among others to find changes in landcover based on aerial imagery (orthophotos, lidar, terrestrial measurements), which consists of new objects that have been added, existing objects that have changed or objects that have been removed in contrast to the latest survey data. As a result of this, this data has to be updated periodically, for example all one to five years, which results in a lot of manual work, as it is often not trivial to find the changes mentioned above. Due to this, the approach of this work is to apply machine learning and a modern GIS as an editing tool. Unfortunately, the current state of the art technologies are still nod advanced enough, to only rely on it when trying to find these changes. Instead, it is just a support of human work, using artificial intelligence, what is called intelligence augmentation. In this work we concentrate on analyzing selected objects/classes from landcover in an orthophoto, and compare this with the current landcover vector data layer.

The result of this work is a QGIS plugin, which sends an image of the current extent together with loaded cadastral survey data to the backend, which then predicts the all objects within the classes *building*, *highways*, *vineyard*, *tennis court* and *swimming pool* using a neural network. In the next step, these predictions are georeferenced and the actual changes are determined by comparing the old vector data from the cadastral survey to the predictions. Finally, the changes are sent back to QGIS and visualized there accordingly to their type (changed, deleted, added). Due to the fact, the network has been trained using satellite imagery from Microsoft Bing and vector data from OpenStreetMap, only about 80% - 90% of the predictions are correct. This is a result from the fact, that the satellite imagery has to go through a photogrammetric post processing and this can lead to displacements regarding the OpenStreetMap vector data.

Acknowledgments

Prof. Stefan Keller tbd

My beloved wife Nadine for supporting me whenever needed by listening, with thoughts, ideas and sometimes by doing a bit of additional housework.

My family for always trying to understand what my thesis is actually about.

Declaration of Originality

I hereby declare that,

1. this thesis and the work reported herein was composed by and originated entirely from me unless stated otherwise in the assignment of tasks or agreed otherwise in writing with the supervisor.
2. all in formation derived from the published and unpublished work of others has been acknowledged in the text and references are correctly given in the bibliography.
3. no copyrighted material (for example images) has been used illicitly in this work

Seengen, 07.06.2018

Martin Boos

Contents

1 Management Summary

Overview 2

2.1 Situation

Periodically, cadastral survey data has to be updated. Generally spoken, this means among others to find changes in landcover based on aerial imagery (orthophotos, lidar, terrestrial measurements), which consists of new objects that have been added, existing objects that have changed or objects that have been removed in contrast to the latest survey data. As a result of this, this data has to be updated periodically, for example all 1-5 years, which results in a lot of manual work, as it is often not trivial to find the changes mentioned above. Due to this, the approach of this work is to apply machine learning and a modern GIS as an editing tool. Unfortunately, the current state of the art technologies are still nod advanced enough, to only rely on it when trying to find these changes. Instead, it is just a support of human work, using artificial intelligence, what is called intelligence augmentation. In this work we concentrate on analyzing selected objects/classes from landcover in an orthophoto, and compare this with the current landcover vector data layer.

2.2 Goals

The goals of this thesis are in general to develop a method which reduces the amount of manual work during the update of the cadastral survey data. This method shall use state of the art deep learning technologies, namely recurrent convolutional neural networks. Additionally, a plugin for a modern GIS tool has to be developed, which can be used by the enduser.

Finally, the goals mentioned above, should be reproducable by anyone and as a result of this, open data like imagery from Microsoft Bing and vector

data from OpenStreetMap has to be used, instead of expensive high resolution orthophotos.

2.3 Chapters

This section gives a brief overview over the following chapters and their contents. Firstly, Chapter 3 is about current technologies and developments in the area of computer vision in general and segmentation in particular. Chapter 4 is an introduction into the artificial intelligence and how it is helpful and used in this thesis. Chapter 5 gives an introduction into convolutional neural networks and their architecture in the context image segmentation. Chapter 6 is about practical / technical challenges, that had to be faced during the work. Following, Chapter 7 is a collection of theoretical and experimental results, like the participation in the crowdAI mapping challenge or the data generation tool *Airtiler*. Chapter 8 is mainly about the QGIS plugin that has been developed in order to enable end users to use the backend for predictions of their data. Finally, Chapter 9 are some thoughts and ideas, what some next steps may be in a work based on this one.

State of the art 3

There are various different techniques and technologies which allow to build neural networks for machine learning purposes. Due to the increasing computation power of current hardware, especially graphic cards, more complex and therefore computationally heavy models can be built. As a result of this, the active research leads to a number of improvements of existing networks or completely new networks each year.

One big challenge in the area of computer vision is instance segmentation, which not only tries to detect occurences of a specific class, but also tries to find single instances of it. In other words, it tries to find the contour. In order to compare various architectures built for such a purpose, there are different challenges published on the internet. One of those leaderboards is from the Microsoft COCO challenge [1], which yearly tries to find the most competitive candidates for different tasks, such as object detection, instance segmentation or keypoint detection.

At the time of this writing, one of the best performing neural networks for instance segmentation is Mask R-CNN [2], which is based on Faster R-CNN [3]. Another well performing architecture can be found with U-Net [4] that has been created for medical purposes. Nonetheless, it can also be used for semantic segmentation of all kind of image data.

4 Introduction

4.1 Intelligence Augmentation

Periodically, cadastral survey data has to be updated. Generally spoken, this means among others to find changes in landcover based on aerial imagery (orthophotos, lidar, terrestrial measurements), which consists of new objects that have been added, existing objects that have changed or objects that have been removed in contrast to the latest survey data. As a result of this, this data has to be updated periodically, for example all one to five years, which results in a lot of manual work, as it is often not trivial to find the changes mentioned above.

One of the most time consuming parts of this work is to find the actual changes, for example a new built house or a garage that has been added to an already existing building. In general, this is a tedious and time consuming task, that can not easily be automated. Due to this, our intention is not to develop a fully automated method but instead to support the person who is doing this work with artificial intelligence, what is called intelligence augmentation [5] and is a rather old concept. However, as described in the previous chapter, the intention is to do intelligence augmentation using state of the art technologies.

To understand the difference between intelligence augmentation (IA) and artificial intelligence (AI) Figure 4.1 shows some examples.

	Intelligence Augmentation ("IA")	Full Automation ("AI")
Enterprise	• Special-purpose learning technologies that automate many mundane tasks at work • Virtual automated assistants • AR glasses used by oil workers, repair professionals, surgeons	• Artificial General Intelligence – machines can perform any task a human can – and better + faster
Autonomous Vehicles	• Automated driving on highways, campuses and in other access controlled situations. Humans needed to handle exceptions and tougher driving conditions, coordinate/operate the overall system	• Networks of fully autonomous vehicles. Society moves away from vehicle ownership and human driving is made illegal • Fully Autonomous vehicles, drones redefine logistics
Robots / Industrial IoT	• Collaborative Robots that work alongside humans and can handle tasks that are hard, unsafe or repetitive • Connected Factories – Predictive Maintenance, Asset Monitoring, Better safety	• Fully automated factories. Robots and automated systems replace all humans on the factory floor
Drones	• Intelligent (but human piloted or limited scope) drones for inspections of cellular towers, building roofs, power cables, remote areas – enabling many new tasks that are unsafe or impossible for humans	• Fully autonomous drones that don't require human supervision, intervention or processing

FIGURE 4.1 IA compared to AI
Source: https://www.cbinsights.com/research/ai-vs-intelligence-augmentation-applications, 02.06.18

4.2 Architecture

Figure 4.2 shows an overview of the architecture. Note, that this diagram shows a combination of the learning and the prediction architecture. Step 2, which is loading the satellite imagery from Microsoft Bing is only required in the learning phase, that is for the generation of the training data, which is required to train the neural network.

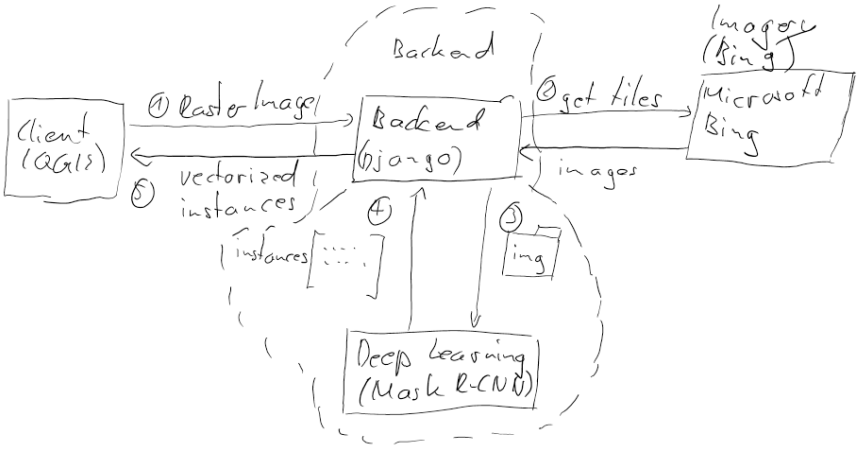


FIGURE 4.2 Architecture overview

Figure 4.3 shows the data flow during the training of the neural network. For this step, satellite imagery from Microsoft Bing maps as well as OpenStreetMap (OSM) data is used. The satellite imagery is downloaded tile per tile for a predefined boundingn box and zoom-level and at the same time, binary images are created from the OSM data, which represent the ground truth. To simplify this step and make it available to the public, a tool called Airtiler [6] has been created and published.

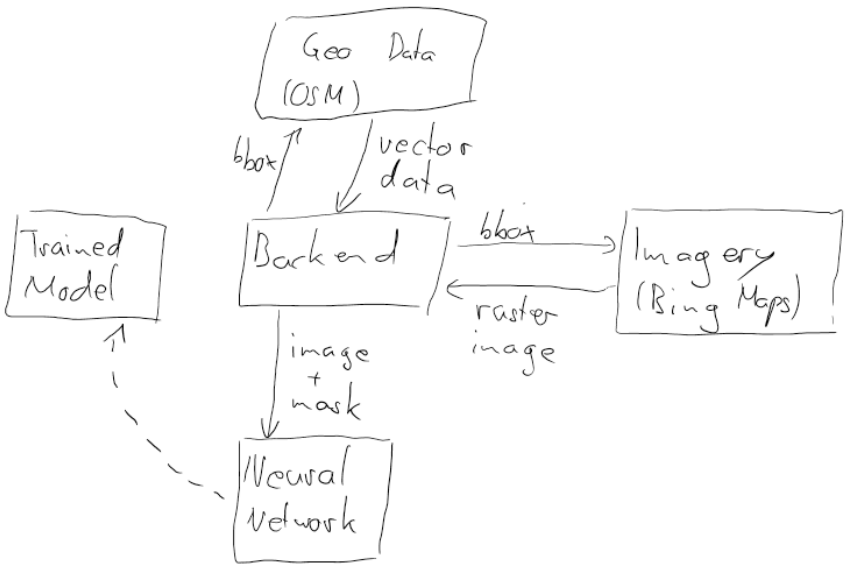


FIGURE 4.3 Learning phase

As soon as the training is completed, the prediction can be done. For this thesis, this has been split into two phases. Figure 4.4 shows the data flow of phase 1, which passes the current extent as base64 encoded image data as well as the bounding box of the current QGIS extent to the configured backend webserver. The pretrained neural network is then used, to predict all instances on the current image. In the next step, the predicted instances are georeferenced using the boundingbox information that was sent to the backend at the beginning of the prediction phase. Finally, the georeferenced instances are sent back to the frontend client (the QGIS plugin), which then visualizes the data on a new layer in QGIS.

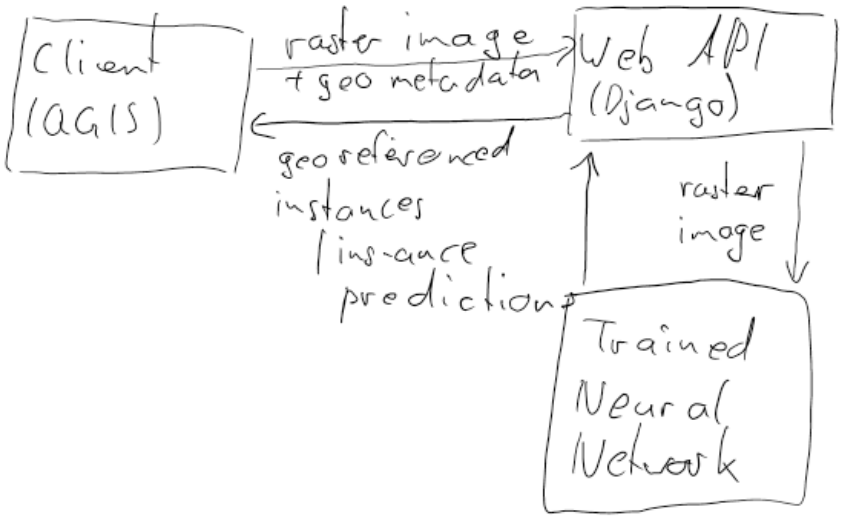


FIGURE 4.4 Prediction phase 1

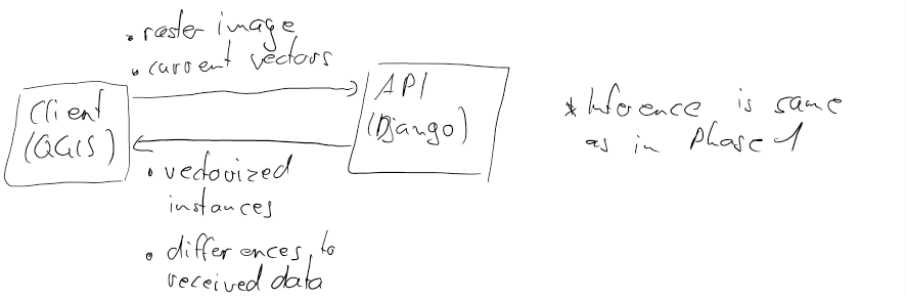


FIGURE 4.5 Prediction phase 2

4.3 Hardware Requirements

To train the neural network, a graphics card is required. We used a Nvidia P100 with 12 gigabytes of memory. A Nvidia GTX 750 was not sufficient due to only 2 gigabytes of memory. Additionally, CUDA[1] has to be installed aswell as Docker[2] and Nvidia-docker[3].

How the docker image can be built and run is described in detail in the readme of the repository[4].

4.4 User Scenario

This section describes a user scenario that can be used for example in a showcase or to instruct the actual end user on how to use the plugin with its corresponding backend. This user scenario requires, that the plugin is correctly installed in QGIS and the backend has been configured correctly in the settings of the plugin. Additionally, the backend has to be running.

1. Start QGIS
2. Load the data of the cadastral survey data. This may for example be a zip archive containing shape files according to the MOpublic [7] specification. In this case, the archive can simply be dragged and dropped into QGIS. Followingly, a dialog will open which allows the select several layers.
3. Select the layers containing buildings and / or other geometries. Typically and according to MOpublic this layer is named *BB_BoFlaeche*.
4. Load an imagery layer, for example Bing maps or Google maps. This can be done easily using the QGIS plugin *OpenLayers*.

[1] https://developer.nvidia.com/cuda-toolkit (22.06.18)
[2] https://www.docker.com/ (22.06.18)
[3] https://github.com/NVIDIA/nvidia-docker (22.06.18)
[4] https://github.com/mnboos/osm-instance-segmentation/blob/master/README.md (22.06.18)

Image Segmentation with Convolutional Neural Networks (CNN) 5

5.1 Introduction

With the increasing computational power that comes with recent graphic cards, increasingly complex neural networks can be used on increasingly challenging tasks. Especially the area of image processing, deep learning gains in popularity. Not only due to the great availability of data sets but also because companies recognize the amount of knowledge and information that can be retrieved with such technologies.

The following sections are a brief introduction into image segmentation using deep learning.

5.1.1 Object detection and segmentation

Object detection exists since long before deep learning was so popular as it is now. In object detection, the goal is to determine whether an object of a specified class (for example 'car') is visible on a image.

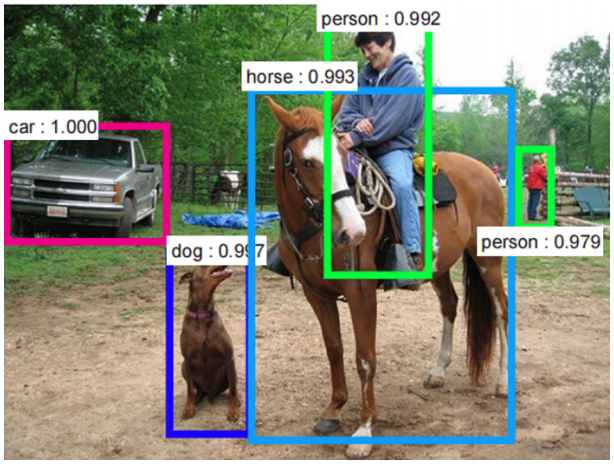


FIGURE 5.1 An example of object detection, showing the detected classes and the confidence of each prediction.
Source: https://dius.com.au/2016/12/06/the-cleverness-of-deep-learning/ (27.05.2018)

In contrast to object detection, the target of image segmentation is not only to state whether an object is present on the image, but to label each pixel of the image with a class, such as 'car' or 'building'. The Different types of image segmentation can be seen in Figure 5.2.

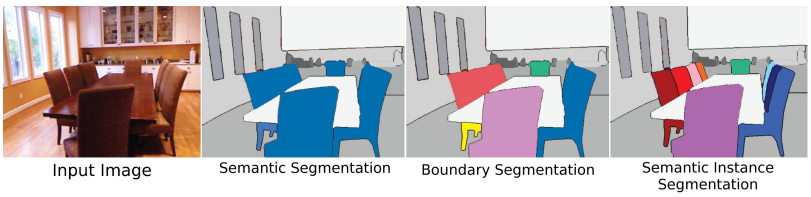


FIGURE 5.2 Different types of image segmentation.
Source: https://i.stack.imgur.com/mPFUo.jpg (27.05.2018)

5.2 Convolutional Layer

A convolutional layer is one of the most important and basic building blocks of a neural network. It has a number of filters, each of which is small, compared to the input volume (the image), for example 5x5x3 pixels (a 5x5 filter with 3 channels, because standard images have 3 color channels). During the forward

pass of the network, the filters are being moved over the input image and at each position of the filters on the image, a convolution is being computed, which is an element wise matrix multiplication and a sum over the resulting matrix. The result of this operation is an activation map, which is also the output of the convolutional layer.

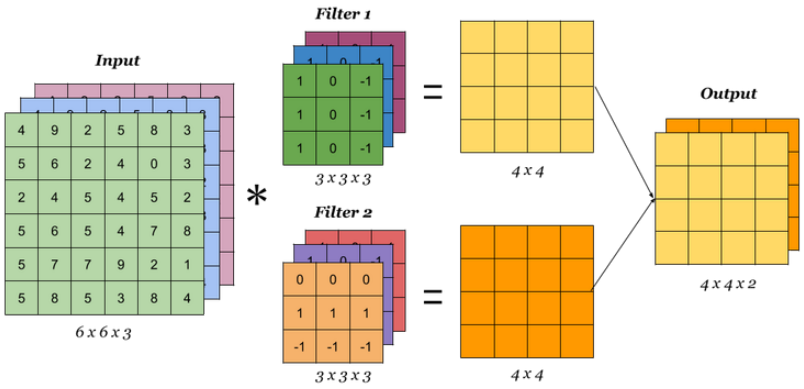


FIGURE 5.3 Convolution with multiple filters
Source: https://idoml.com (03.06.2018)

Obviously, the size of a filter can be configured, as well as the step size, the stride, and the amount of zero padding around the input image.

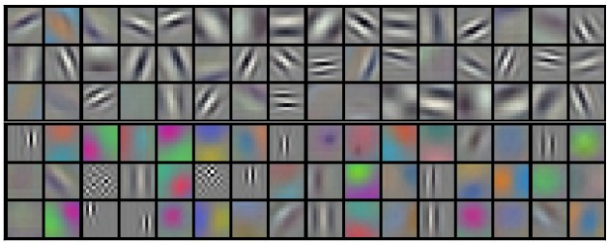


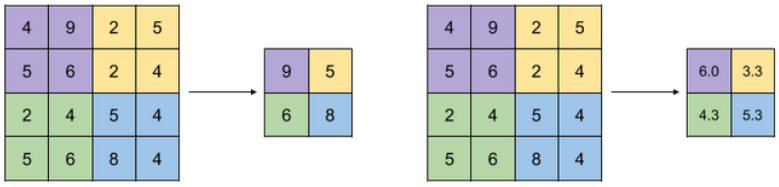
FIGURE 5.4 Example filters learned by [8].
Source: http://cs231n.github.io/assets/cnn/weights.jpeg (27.05.2018)

5.3 Pooling Layer

Pooling is a technique which allows to reduce the size of an image by extracting a single value from a region of values. The extracted value depends on the

pooling type that is used. The most common pooling types are max pooling for extracting the highest value from the current field and average (avg) pooling which extracts the lowest value.

A pooling layer has basically trhee parameters, it can be configured with. Firstly, there is the stride s which is the distance the filter is moved. Secondly, there is the filter size f which determines the width and height of the filter that is used, to extract the value from the input.



(A) Max Pooling (B) Average Pooling

FIGURE 5.5 Max and average pooling
Source: https://idoml.com (02.06.2018)

As described in the previous chapter, a layer can have multiple output channels. Obviously, pooling can also be done with multiple channels, which can be seen in Figure 5.6.

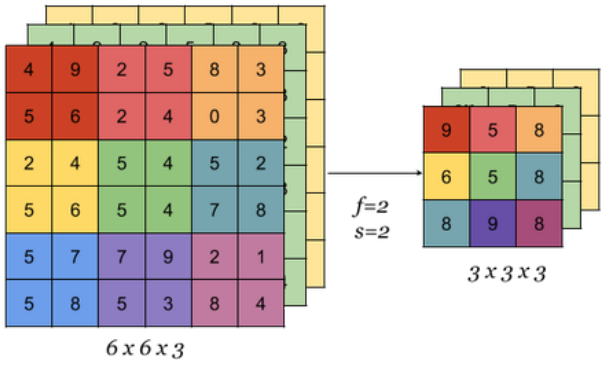
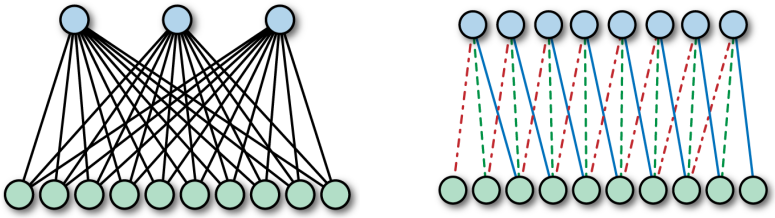


FIGURE 5.6 Max pooling with multiple channels
Source:https://idoml.com (03.06.2018)

5.4 Fully Connected Layer

A fully connected layer is nearly the same as a convolutional layer. The only difference is, that the output of a convolutional layer is spatially only connected to a region of the previous layer but not to the whole output. Figure 5.7 shows the difference between a fully connected layer and a convolutional layer.



(A) Fully Connected Layer (B) Convolutional Layer

FIGURE 5.7 Fully Connected and Convolutional Layers
Source: https://www.safaribooksonline.com/library/view/learning-tensorflow/9781491978504/ch04.html (07.06.2018)

5.5 Mask R-CNN

Due to the massive amount of research required to come up with or improve an existing state of the art neural network architecture, this has been out of scope for this work. As a result of this, we decided to use Mask R-CNN [2] which is, as the name implies, a recurrent, convolutional neural network.

Practical Challenges 6

6.1 Training data

When it comes to data sets that can be used when training the neural networks, there are two options. Either one uses an already available data set, either paid or free, or a new data set is created. Currently, machine learning and especially deep learning are popular topics. As a result of this, the availability of free and publicly available data sets has increased, especially in the area of image processing like segmentation, facial recognition or object detection. Free data sets consisting of aerial imagery are [9], [10], [11], [12], [13], [14].

Despite these available data sets, we decided to make our own, consisting solely of open data, that is Microsoft Bing for the imagery and OpenStreetMap for the vector data. Due to this, a tool named Airtiler [6] which is described in detail in Chapter 7.

It can be assumed, that in the future, more and more swiss cantons will made high resolution orthophotos publicly available. At the time of this writing, especially the canton of Zurich takes a pioneering role and makes several of their data sources publicly and freely available[1]. However, at the time of this writing, it was not an option to use these images for this work, because it would lead to a rather small dataset.

[1] https://geolion.zh.ch/ (15.06.18)

6.2 Prediction accuracy

6.2.1 Class probability

After the first training the neural network, the results were not quite as expected. Even though, buildings were predicted as buildings in most cases, other classes, like tennis courts, were predicted as buildings as well. Due to this, the network has been retrained with the additional, incorrectly predicted, classes like tennis courts. However, instead of correctly making a distinction between buildings and tennis courts, the overall prediction accuracy got worse. This might be the result of the network which has to solve a more complex task now, by deciding which class it is, instead of a simple yes-no decision. Additionally, the training data is highly imbalanced, as there are lot more samples of buildings than tennis courts. As a result of this, a solution could be to train the network several times seperatly, to get multiple models, each trained for a specific class. Another solution could be to weight the loss according to the relative amount of the specific class according to the size of the whole dataset.

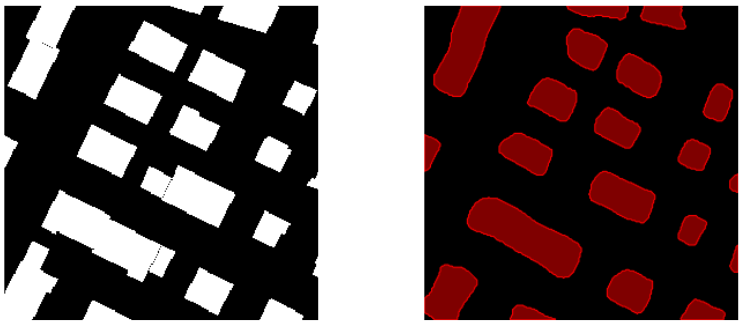
6.2.2 Outline

Figure 6.1 shows, that the predictions are in most cases a bit to small when compared to the corresponding orthophoto. This might be the result of slightly misaligned masks, since the masks and the images are generated seperately.

FIGURE 6.1 Too small predictions

6.3 Building outline regularization

Once the network has been trained, it can be used to make predictions on images, it has never "seen" before. However, there are situations in which the predictions are far from perfect, especially then if the building is partially covered from trees or has unclear outlines. This can be seen in Figure 6.2.



(A) Actual ground truth (B) Predicted building masks

FIGURE 6.2 Predicted masks and actual ground truth

As a result of the inaccuracies in the predicted building masks the contours of these predictions can not directly be used to create the vectorized outlines. Instead, the predictions have to be regularized. The approach used is similar to [15] and described in Figure 6.3.

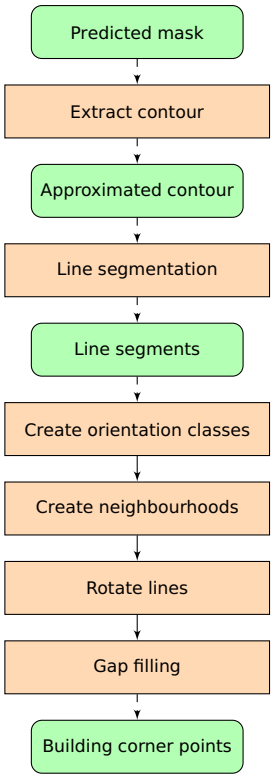


FIGURE 6.3 Rectangularization procedure

The single steps are described in detail in the following sections.

6.3.1 Contour extraction

The first step of the building outline regularization procedure consists of getting the contour from the predicted mask, which covers the whole building. The extraction is done using the marching squares algorithm [16]. In this algorithm, a square consisting of four cells is moved (marched) along the contour in such a way, that at least one cell always covers the object to be contoured. Additionally, the square always has a state, which is derived from the content of its cells, according to (6.1). The cells are traversed in counter clockwise order.

$$
\begin{aligned}
s &= \sum_{i=0}^{3} 2^i f(c_i) \\
&= 2^0 * f(c_0) + 2^1 * f(c_1) + 2^2 * f(c_2) + 2^3 * f(c_3) \\
&= f(c_{bottomLeft}) + 2 * f(c_{bottomRight}) + 4 * f(c_{topRight}) + 8 * f(c_{topLeft})
\end{aligned}
\tag{6.1}
$$

where:

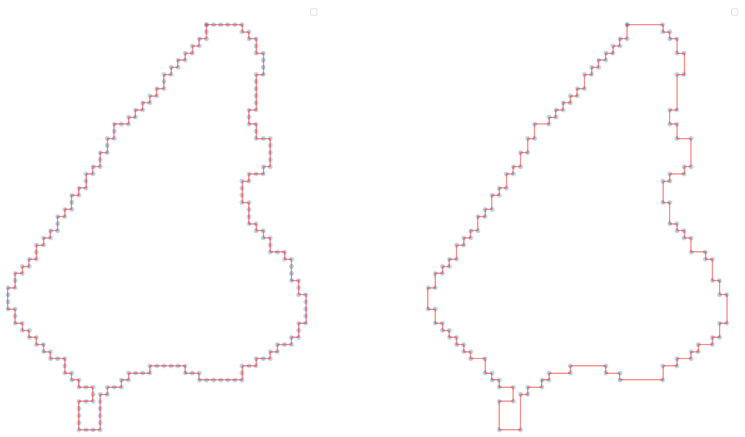
c_i: The value of the cell i

c_0: The bottom left cell

and

$$
f(c_i) = \begin{cases} 0 & \text{if } c_i \leq 0 \\ 1 & \text{if } c_i > 0 \end{cases}
$$

As soon as the contour has been extracted, its number of points will be reduced using a Douglas-Peucker algorithm [17]. The reason for this is, that the contour has pixel accuracy. That means, there may be several points on the same horizontal or vertical line, even though, the startpoint and endpoint of each such line would be enough, to represent the line. Additionally, the lower the number of points per contour is, the faster the following processing will be.



(A) Original contour (B) Approximated contour

FIGURE 6.4 Contour before and after approximation

6.3.2 Line segmentation

Once the contour has been extracted, it is split into multiple line segments. For this, the main direction of the building is determined using the Hough Transformation [18]. The result of the Hough Transformation is an datastructure, which contains an angle, a distance and a number. The combination of the angle and the distance, from a predefined reference point, lead to a line. The number is the number of points which lie on the constructed line. Therefore, this algorithm can be used to detect the main building orientation, i.e. the longest contour-line of any orientation. The angle of the found line, is called the main building orientation.

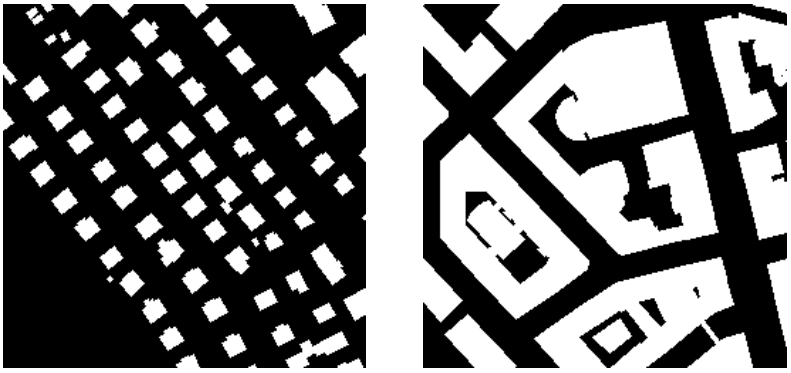
Once the main building orientation is known, the line segmentation starts at the point which has the smallest distance to this line. The whole procedure is depicted in algorithm 1.

```
Data: Contour points, startpoint
Result: Lines
rearrange points so that startpoint == points[0]
lines = []
while any points left do
    segment = remove first 3 elements from points
    while points is not empty do
        p = points.first()
        err = root mean square error of distance between segment.last() and
         p
        if err > threshold then
            break
        end
        segment.append(p)
        points.remove(p)
    end
    if segment.length() >= 3 then
        line = fit line to points of segment
        lines.append(line)
    end
end
```

Algorithm 1: Line segmentation

6.3.3 Create orientation classes

After the line segmentation, an orientation will be assigned to each line. Generally, most of the buildings consist of mainly right angles. However, there are still buildings, for which this assumption is not true. Due to this, orthogonality will be preferred, but other angles will still be possible. Algorithm 2 shows the procedure, which assigns a main orientation class to each line and defines the lines parallelity / orthogonality to the longest line of the same orientation class.



(A) Regular buildings (B) Non-orthogonal buildings

FIGURE 6.5 Different kinds of buildings with regard to their corner angles

Data: Lines, angleThreshold
while *any line without orientation* **do**
  line = longest of unprocessed lines
  line.orientation = angle between line and horizontal-line
  foreach *line li without orientation* **do**
    a = angle between line and li
    if $a \leq$ *angleThreshold* **then**
      li.orientation = line.orientation li.orthogonal = line.orthogonalTo(line)
    end
  end
end

Algorithm 2: Orientation assignment

6.3.4 Create neighbourhoods

At this point, each line segment belongs to a orientation class and the parallelity / orthogonality of each line to the orientation classes main line is known. However, the spatial location of each line has not been taken into account yet, which is done in this step. Clusters of neighbouring lines are created within each orientation class. As a result of this, it is now possible to find lines, which may be better placed in an other orientation class. This is based in the assumption, that it is unprobable, that a line k of the orientation class x is surrounded by lines of the orientation class y. In this case, the line k will be assigned to the orientation class y.

6.3.5 Update line orientation

Finally, the lines will be adjusted to their orientation class with respect to each lines parallelity / orthogonality. The result of such an adjustment, can be seen in Figure 6.6, which shows a single orientation class and lines, which have been adjusted either parallel or orthogonal to the orientation class.

FIGURE 6.6 Adjusted lines

6.3.6 Gap filling

In order to create the final building outline, the only thing left to do is to fill the gaps between the line segments.

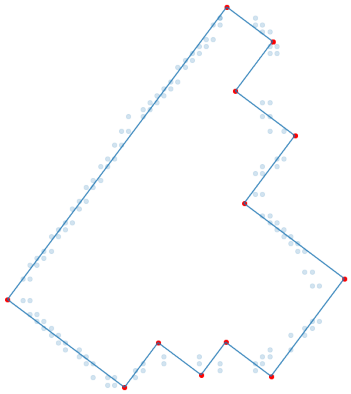


FIGURE 6.7 Final building outline

Theoretical and Experimental Results 7

7.1 Training data

7.1.1 Airtiler - A data set generation tool

For the training, we wanted to use publicly and freely available data. Not only due to the fact, highly resolved orthophotos cost quite a lot but also to make it possible for others to reproduce the results.

As a result of this, OpenStreetMap was chosen for the vector data and Microsoft Bing Maps for the imagery. A dataset consisting of satellite imagery and images for the ground truths can be created using the Python module Airtiler [6]. This tool has been developed by the author during this master thesis. It allows to configure one or more bounding boxes together with several other options like zoom level and OpenStreetMap attributes.

Listing 7.1 shows a sample configuration as it is being used by Airtiler.

```
{
  "options": {
    "target_dir": "",
    "zoom_levels": [18,19],
    "separate_instances": false
  },
  "query": {
    "tags": [
        "highway",
        "building",
        "leisure=swimming_pool",
        "sport=tennis",
        "landuse=vineyard"
    ]
  },
  "boundingboxes": {
    "giswil_1": [
        8.1566193073,
        46.7783248574,
        8.1681635349,
        46.7905387509
    ],
    "giswil_2": [
        8.1388001434,
        46.7778439103,
        8.1562840923,
        46.7880064196
    ],
    "goldach_rorschacherberg": [
        9.440673825,
        47.4534664711,
        9.5168056457,
        47.4896115582
    ]
  }
}
```

Listing 7.1 Sample configuration for Airtiler

In the next step, the configured bounding boxes are iterated and for each box

all tiles, according to the Tile Map Service (TMS) specification [19] are being processed. This includes the download of the corresponding orthophoto from Bing Maps, as well as the creation of at least one binary image, which contains all ground truth instances of the current class, for example *building* or *highway*, that have been specified in the configuration. As a result of this, one satellite image and one or more binary images are created per tile. If for the current tile no ground truths according to the configuration are found, no images for this tile will be created. Figure 7.1 shows the output of a single tile, generated by Airtiler. One important point are the filenames, which are generated according to the content of the current file.

(A) 18_138079_170377.tiff

(B) 18_138079_170377_building.tif

(C) 18_138079_170377_highway.tif

(D) 18_138079_170377_pool.tif

FIGURE 7.1 Output of Airtiler for the TMS tile (x,y)=(138079,170377) at zoom level 18. The file names are generated according to the content, which makes the loading and interpretation of the data set rather simple; Images are named with the file extension *.tiff* whereas the masks have the extension *.tif*

# Epochs	# Steps / Epoch	# Validation Steps	Config Change	AP@0.5	AR@0.5
100	2500	150	-	0.798	0.566
100	2500	150	Image mean RGB updated	0.799	0.564
100	2500	150	Mini mask disabled	0.807	0.573
100	5000	200	+ validation steps	0.821	0.599
100	10000	200	+ steps / epoch	0.833	0.619
100	20000	300	+ Validation steps, + steps / epoch	0.853	0.885

TABLE 7.1 Mapping challenge results

7.1.2 Publicly available data sets

Furthermore, there are several different datasets publicly available: [9], [10], [11], [12], [13], [14].

7.2 Mapping Challenge

At the time of this writin the platform crowdAI hosted a challenge called Mapping Challenge [20] which was about detecting buildings from satellite imagery. In order to gain additional knowledge regarding the performance of Mask R-CNN, we decided to participate in the challenge.

Table 7.1 shows the changes made to the Mask R-CNN config and their impact on the prediction accuracy.

Generally, these results indicate, that finetuning of hyperparameters has an impact. Furthermore, the even bigger impact can be made just by longer training. However, in case of longer training, one has to make sure, that the network will not overfit. In the case of an already existing architecture like Mask R-CNN for example, this has already been done. On the other hand, if one develops a new architecture, overfitting has to be taken care of, for example with a technique called Dropout [21]. Generally, Dropout randomly disables some units during the training. As a result of this, the model constantly changes and overfitting can not happen that easily.

7.3 Microsoft COCO Annotation Format

For the crowdAI Mapping Challenge [20] the instances were represented in the Microsoft COCO annotation format [22]. Unfortunately, using this format, it is not possible to represent polygons with holes in it[1]. As there are many buildings, for which this would be required, we decided not to use this format and instead use images for the representation of the ground truths.

(A) A building with multiple holes (B) Predicted building masks

FIGURE 7.2 The corresponding ground truth

7.4 Building detection

tbd

[1] https://github.com/cocodataset/cocoapi/issues/153,21.05.2018

8 Practical Results

8.1 QGIS Plugin

A QGIS plugin has been created, which allows to process the currently displayed imagery in a corresponding backend server. Figure 8.1 shows an image of the imagery layer overlayed with the changes generated by the backend. The red dots on the upper left indicate deleted objects, or in other words, buildings that were not predicted in the image but were existent at the same location in the cadastral survey data. However, it is obvious, that in this case the neural network is wrong, as the actual buildings can be seen clearly. Additionally, in the lower middle the blue area can be seen, which indicates a change in the survey data.

A prediction is classified as change, as soon as it fully covers at least one existing object in the survey data. Figure 8.2 shows the same changes on the cadastral data layer and Figure 8.3 shows the predictions as returned by the neural network.

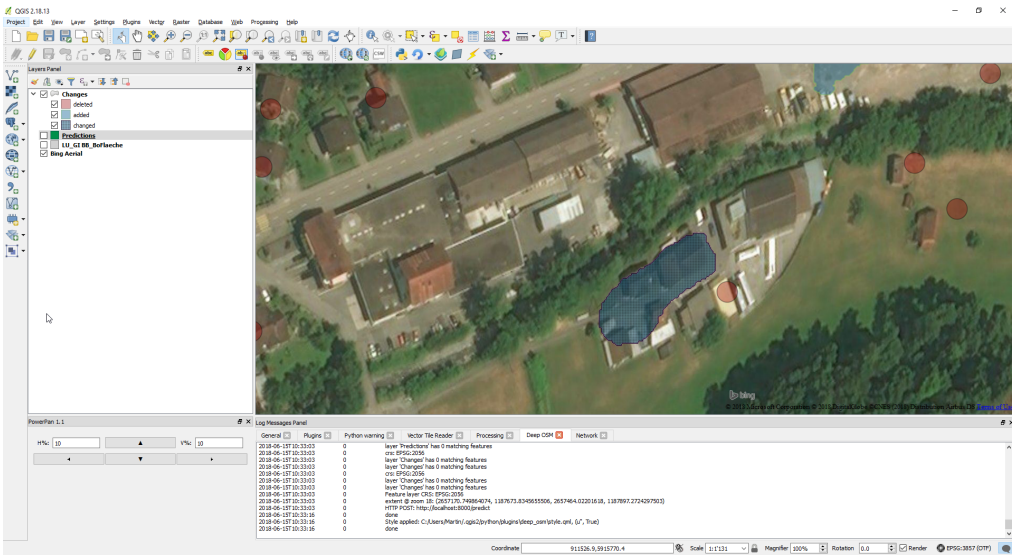


FIGURE 8.1 Changes in QGIS

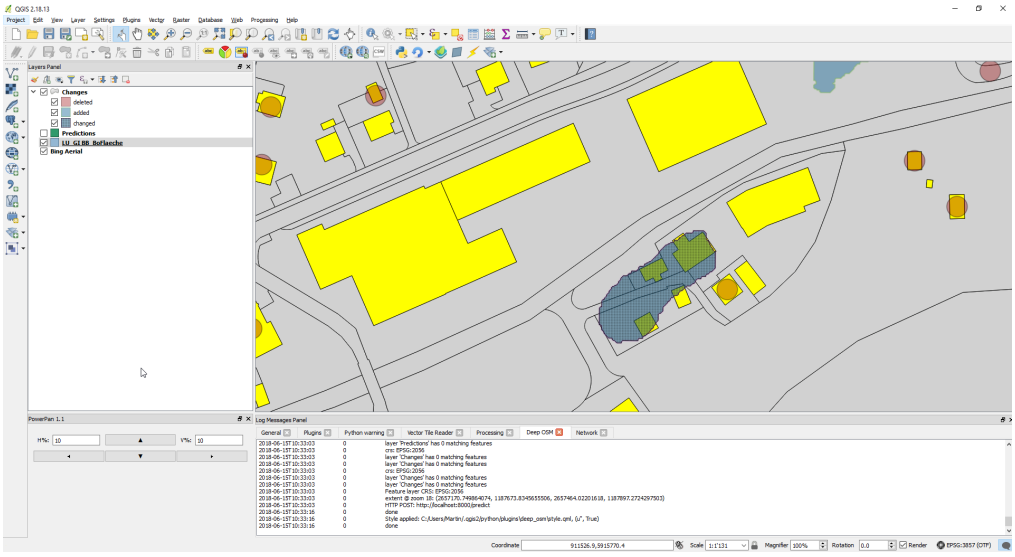


FIGURE 8.2 Changes on cadastral survey data layer

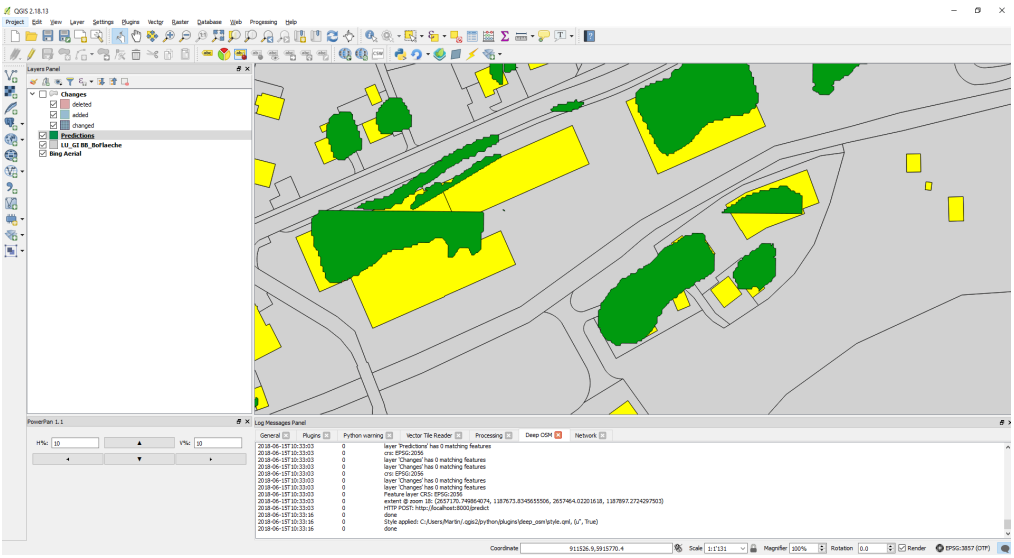


FIGURE 8.3 Predictions (green) as generated by the neural network

FIGURE 8.4 Predictions have attributes showing the predicted class (building in this case)

FIGURE 8.5 Changes have attributes showing the predicted class and the type of change (added, deleted, changed)

Conclusion and Future Work 9

9.1 Training Data

Assumingly, a more accurate and less imbalanced data set would probably lead to a higher prediction quality. Using higher resolved images, it might even be possible to let the neural network learn the differences between asphalt and gravel roads. Additionally, the predicted masks would probably also be more accurate and less wrong predictions would happen. Generally spoken, there might be an increase in the overall prediction quality.

tbd: training with lower zoom level, as we're only interested in the changes and not the actual contours.

9.2 QGIS Plugin

Instead of sending an image and corresponding vectors to the backend, an option could be, to send two images, an old and a new one, to the backend. All changes can then be derived from the differences in the predictions of those two images. However, often one does not have access to either old or up-to-date imagery, which is a problem for the use case described in the introduction.

Another improvement would be to give the user the possibility, to manually choose a perimeter, which is then automatically processed, even though it might take a longer time.

Appendices

List of Figures

List of Tables

Bibliography

[1] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, *Microsoft coco: Common objects in context*, 2014. [Online]. Available: http://arxiv.org/pdf/1405.0312.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn*, 2017. [Online]. Available: http://arxiv.org/pdf/1703.06870.

[3] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, 2015. [Online]. Available: http://arxiv.org/pdf/1506.01497.

[4] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. [Online]. Available: http://arxiv.org/pdf/1505.04597.

[5] D. Engelbart, „Augmenting human intellect: A conceptual framework“, 1962.

[6] Martin Boos, *Airtiler*. [Online]. Available: https://github.com/mnboos/airtiler.

[7] *Mopublic*, 16.06.2017. [Online]. Available: https://www.cadastre.ch/content/cadastre-internet/de/manual-av/service/mopublic/_jcr_content/contentPar/tabs_copy_copy/items/dokumente/tabPar/downloadlist/downloadItems/102_1472647336994.download/Weisungen-MOpublic-de.pdf.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, „Imagenet classification with deep convolutional neural networks“, 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[9] Volodymyr Mnih, „Machine learning for aerial image labeling“, Dissertation, 2013. [Online]. Available: https://www.cs.toronto.edu/~vmnih/data/.

[10] *Spacenet on amazon web services (aws)*, 30.04.2018. [Online]. Available: https://spacenetchallenge.github.io/datasets/datasetHomePage.html.

[11] International Society for Photogrammetry and Remote Sensing, *Vaihingen*. [Online]. Available: http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-vaihingen.html.

[12] International Society for Photogrammetry and Remote Sensing, *Potsdam*. [Online]. Available: http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html.

[13] P. Helber, B. Bischke, A. Dengel, and D. Borth, *Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification*, 2017. [Online]. Available: http://arxiv.org/pdf/1709.00029.

[14] *Deepsat*. [Online]. Available: http://csc.lsu.edu/~saikat/deepsat/.

[15] T. Partovi, R. Bahmanyar, T. Kraus, and P. Reinartz, „Building outline extraction using a heuristic approach based on generalization of line segments“, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 3, pp. 933–947, 2017, ISSN: 1939-1404. DOI: 10.1109/JSTARS.2016.2611861.

[16] C. Maple, „Geometric design and space planning using the marching squares and marching cube algorithms“, in *2003 international conference on geometric modeling and graphics*, E. e. Banissi and M. Sarfraz, Eds., IEEE Comput. Soc, 2003, pp. 90–95, ISBN: 0-7695-1985-7. DOI: 10.1109/GMAG.2003.1219671.

[17] D. H. Douglas and T. K. Peucker, „Algorithms for the reduction of the number of points required to represent a digitized line or its caricature“, *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973, ISSN: 0317-7173. DOI: 10.3138/FM57-6770-U75U-7727.

[18] R. O. Duda and P. E. Hart, „Use of the hough transformation to detect lines and curves in pictures“, *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972, ISSN: 0001-0782. DOI: 10.1145/361237.361242. [Online]. Available: http://dl.acm.org/ft_gateway.cfm?id=361242&type=pdf.

[19] *Tile map service specification*. [Online]. Available: https://wiki.osgeo.org/wiki/Tile_Map_Service_Specification.

[20] crowdAI, *Mapping challenge*. [Online]. Available: https://www.crowdai.org/challenges/mapping-challenge (visited on 04/29/2018).

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, „Dropout: A simple way to prevent neural networks from overfitting“, *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014, ISSN: 1532-4435. [Online]. Available: http://dl.acm.org/ft_gateway.cfm?id=2670313&type=pdf.

[22] *Coco data format: Common objects in context*. [Online]. Available: http://cocodataset.org/#format-data.