ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume IV-2/W5, 2019
ISPRS Geospatial Week 2019, 10–14 June 2019, Enschede, The Netherlands

FACADE RECONSTRUCTION FOR TEXTURED LOD2 CITYGML MODELS BASED ON DEEP LEARNING AND MIXED INTEGER LINEAR PROGRAMMING

S. Hensel[1,*], S. Goebbels[1], M. Kada[2]

[1] Institute for Pattern Recognition, Niederrhein University of Applied Sciences,
Reinarzstraße 49, 47805 Krefeld, Germany - (Simon.Hensel, Steffen.Goebbels)@hsnr.de
[2] Institute of Geodesy and Geoinformation Science, Technical University of Berlin,
Straße des 17. Juni 135, 10623 Berlin, Germany - Martin.Kada@tu-berlin.de

KEY WORDS: Facade Reconstruction, Faster R-CNN, Mixed Integer Linear Programming, CityGML

ABSTRACT:

The paper describes a workflow for generating LoD3 CityGML models (i.e. semantic building models with structured facades) based on textured LoD2 CityGML models by adding window and door objects. For each wall texture, bounding boxes of windows and doors are detected using "Faster R-CNN", a deep neural network. We evaluate results for textures with different resolutions on the ICG Graz50 facade dataset. In general, detected bounding boxes match very well with the rectangular shape of most wall openings. Thus, no further classification of shapes is required. Windows are typically aligned to rows and columns, and only a few different types of windows exist for each facade. However, the neural network proposes rectangles of varying sizes, which are not always aligned perfectly. Thus, we use post-processing to obtain a more realistic appearance of facades. Window and door rectangles get aligned by solving a mixed integer linear optimization problem, which automatically leads to a clustering of these openings into few different classes of window and door types. Furthermore, an a-priori knowledge about the number of clusters is not required.

1. INTRODUCTION

CityGML is the standard description language for semantic city models (Gröger et al., 2012) and is used for cadastral purposes but also for many different simulation tasks, see (Biljecki et al., 2015). Building models can be given in different levels of detail (LoD). In LoD2, a building model simply is composed of wall polygons, roof polygons and a ground plane. Walls and roofs can be textured. Based on such textures, we detect and add window and door objects to CityGML models. Since windows and doors can only be modeled in LoD3 or higher, we transform textured LoD2 models into their corresponding LoD3 versions, i.e., we map textures to geometries.

Facades can be analyzed in 2D based on images or in 3D based on point clouds. If high quality laser scanning point clouds are available, rectangular windows can be found quite easily: An oriented facade's plane can be taken from a city model and then one can search for points with a significant distance to this plane, cf. (Tuttas , Stilla, 2013). Since the laser beam passes glass, points behind the plane indicate openings (voyeur effect). The precision is even sufficient to distinguish curtains from facades. The left part of Figure 1 shows a laser scanning point cloud on which rectangular openings were detected, see Figure 2. Unfortunately, facades are only sparsely covered in airborne laser scanning data, and terrestrial laser scanning data might not be available on a large scale. This is different for oblique aerial images. Many LoD2 CityGML models were textured with such images and can be used as input for our proposed workflow. But one could also consider detecting facades based on sparse photogrammetric point clouds derived from oblique aerial images. However, in contrast to high resolution point clouds as discussed in (Malihi et al., 2018), such clouds can become noisy with wave-like artifacts (see Figure 1) and do not

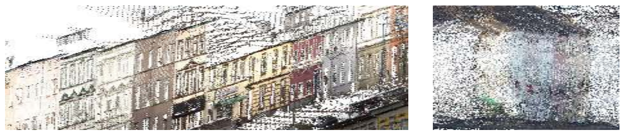
Figure 1. Facades in a terrestrial laser scanning point cloud (left) vs. facades in a photogrammetric point cloud (right)

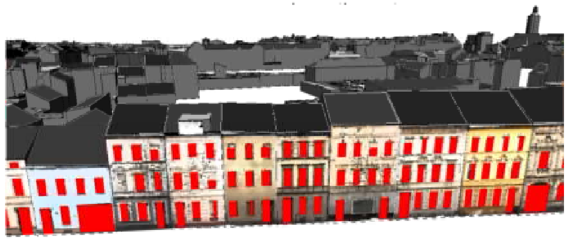
Figure 2. Detected windows and doors based on a terrestrial laser scanning point cloud in connection with a 3D city model

contain differentiated depth information. Also, windows might not appear as holes in photogrammetric clouds. Thus, we focus on window and door detection based on texture images in this paper. There might be shadows on facades but detection and removal of shadows in complex images still is challenging, see (Finlayson et al., 2002), (Guo et al., 2011). Also, parts of images might be occluded by other buildings or vegetation. In this paper, we exclusively work with images that do not show such artifacts. By working with textures, we can exclude perspective distortions and can assume that occlusions by other buildings have been resolved based on the CityGML model.

*Corresponding author

By evaluating horizontal and vertical projection profiles of the facade's grayscale image, regular arranged windows can be easily detected, see for example (Lee , Nevatia, 2004), (Kulkarni et al., 2011), (Miljanovic et al., 2012). However, such an approach is not suitable for irregular or historic facade layouts. Therefore, we apply a more general approach. Section 2 gives an overview of techniques that come into question. We decided to apply the faster region-based convolution neural network (Faster R-CNN) from (Ren et al., 2015), see Section 3. This network has been applied to many object detection and segmentation tasks and provides the position of a bounding box for each detected object in an image and matches the object to a class. Not every pixel within the bounding box has to belong to the classified object, but this can be neglected for rectangular windows and doors. Such objects do match with their bounding boxes quite well. Faster R-CNN's region proposal network (RPN) was already successfully applied to window detection in "DeepFacade", see (Liu et al., 2017). The authors customized a deep convolutional neural network for pixel classification by choosing a problem specific loss function. They set up the function to favor the detection of rectangular symmetric structures. Then pixel based classifications are refined with bounding boxes, which are generated by the RPN of Faster R-CNN (based on a VGG16 network (Simonyan , Zisserman, 2014)). Since Faster R-CNN is used to refine an existing classification, the classification part of Faster R-CNN is not used in their workflow. In contrast to this, we directly apply Faster R-CNN to window and door segmentation as well as to classification. Thus, the initial classification step used in (Liu et al., 2017) can be omitted. To get an optimal layout, we formulate a mixed integer linear optimization problem that can be solved with standard tools, see Section 4. An advantage of this post-processing step is that in contrast to adjusting a loss function, one can easily add further constraints without any implications on detection results. For example, an important additional constraint is that only a small number of different window sizes should occur.

Recently, in (Rahmani , Mayer, 2018) a workflow combining a pre-trained RPN with a structured random forest approach (cf. Section 2) was introduced. However, we obtain better detection results. The reason is that we trained Faster R-CNN with the problem-specific large CMP database, see Section 3. Also, the combined method does not consider architectural rules like window and door alignment.

2. RELATED WORK

In the literature, a wide variety of methods for facade parsing and processing exists. The basic architectural rules that apply to a large number of facades, allow the use of model based approaches like grammars. Grammatical rules can be predefined using human knowledge of facades or learned from real data. For example, in (Wan , Sharf, 2012) the authors use a grammar based algorithm on pre-segmented objects from LiDAR scan layers to reconstruct buildings in 3D. Often, a facade's layout follows a standard pattern. In this case, it is likely that a matching set of grammar rules exists. Then a facade's pattern can be continued in noisy or occluded areas. But, if the facade architecture does not follow a regular pattern, the results possibly will not match reality. Our data are almost free of occlusions. Instead of applying general grammar rules, we therefore focus on aligning windows and doors.

Machine learning techniques are powerful tools for object detection or segmentation in facade images. They do not require model knowledge but a large amount of training data to achieve good results. Conditional Random Field (CRF), a probabilistic graphical model, is one of these techniques. A fully-connected CRF for semantic segmentation of city view images is used in (Li , Yang, 2016). A disadvantage of this approach is high computational complexity, which makes it difficult to re-train the model with new data. In comparison, a random forest method classifies each pixel or point independently, which results in a noisy labeling of the input image. Rahmani, Huang, and Mayer enhance the training of a random forest with architectural constraints and an object hierarchy in (Rahmani et al., 2017). The authors call their enhancement "structured random forest". Similarly, in (Martinović et al., 2015) a random forest classifier in combination with stochastic context-free grammars is used. The aim is to generate 3D-Models of facades using point clouds as input. However, this method fails at making predictions with rare outcomes, as the algorithm is based on bootstrap sampling. Bootstrap sampling is a stochastic model, which can be used for image segmentation. Gadde et al. found a way to improve segmentation results of classifiers on facades using auto-context features, see (Gadde et al., 2018). Auto-context features are gained from the results of a trained classifier and are used to train the classifier of the next iteration. This process is repeated until the results have a sufficient quality. This method can deliver good segmentation results, but involves a great deal of effort due to the number of independent classifiers that have to be trained. A good overview of some common machine learning methods gives the work (Mathias et al., 2016), in which the methods are compared with respect to the paper's own three-layered approach.

Deep learning is motivated by the functionality and capacity of the human brain. Especially the field of facade parsing becomes increasingly popular for benchmarking new network architectures. Similar to our approach, in (Liu et al., 2017) Faster R-CNN and in (Rahmani , Mayer, 2018) an RPN is used for the segmentation of facades, see Section 1. Another improvement of Faster R-CNN was proposed in (He et al., 2017): "Masked R-CNN" is a combination of two already existing state-of-the-art models, an RPN and a binary mask classifier. Within bounding boxes, a classified object is labeled as it is the case with semantic segmentation.

After submission of our work, the paper (Xingzi Zhang et al., 2019) was published. The authors apply an RPN in connection with Mask R-CNN to detect rectangular windows in textures of LoD2 models, align them and then interactively generate LoD3 CityGML models. However, in contrast to our work, different model data are used, doors are not detected automatically and alignment is done by a heuristics and rule based clustering.

Our work focuses on detection using 2D images, but deep learning architectures exist which benefit from 3D information in point clouds. The big problem is that, unlike images which have their structure predetermined by the pixel grid, raw 3D point clouds do not have structure. The input has to be properly structured so that the neural network is able to process it. Many ways exist to overcome this problem, one is to apply structure to 3D data using a voxel grid. This idea was used in (Wu et al., 2015) to introduce 3D Shapenets. The authors translate multi-view 2.5D depth maps into voxel representations, which serve as input for a CNN that classifies and autocompletes the 3D object. Networks like PointNet++ (Qi et al., 2017) are capable of using 3D point clouds directly as an input. PointNet++ is the successor of PointNet, which had issues if applied on data with a non-consistent point density. PointNet++ addresses this problem by extracting spherical point neighborhoods and applying PointNet on each hierarchical level. Another network for processing point cloud data is SPLATNet (Su et al., 2018). It uses sparse bilateral convolutional layers as building blocks.

3. WINDOW AND DOOR DETECTION USING FASTER R-CNN

The basis of Faster R-CNN consists of a feature map generated by a fully convolutional neural network (FCN). A FCN, originally introduced in (Long et al., 2015) to solve image segmentation problems, consists of convolutional, pooling and upsampling layers. Due to the convolutional layers, which implement only local connections and shared weights, its input is not restricted to a specific size. These filters and even the decision-making layers at the end, are learned during training. Typically, a fully-connected (FC) layer can be combined with a FCN. This would enhance its learning capabilities from using only local information to using global information. In image segmentation, the output has to be upsampled to fit the size of the original input to get the final result. This can be done by using layers that serve as bilinear filters or transposed convolutions.

R-CNNs do not use the whole image as an input for classification, instead region-proposals are used, which tell the network where to look. The original R-CNN was introduced in (Girshick et al., 2014) and uses a selective search algorithm to generate its region proposals. Selective search is able to generate region proposals by computing hierarchical grouping of similar regions based on color, texture, size and shape similarity. At the end, a CNN is applied to these region proposals for the final classification. In (Girshick, 2015) the R-CNN was improved to Fast R-CNN where a technique called Region of Interest (ROI) pooling was introduced. This technique allows for sharing computations during training by reusing the same feature map for multiple ROIs. Generation of region proposals through selective search, as applied in R-CNN and Fast R-CNN, is time-consuming due to the calculation of similarity features between all input image regions. Faster R-CNN resolves this problem by using an RPN, which shares computation with the classification part through reusing the same feature map produced by a FCN. The architecture of the FCN is replaceable, popular architectures include VGG16 (Simonyan , Zisserman, 2014) and ResNet (He et al., 2016).

For applying Faster R-CNN on window and door detection, training and evaluation based on images with a known ground truth had to be prepared. For training, the CMP facade database from (Tyleček , Šára, 2013) was used. This database includes 606 facade images with 12 specified classes, where only window and door elements were selected for our purpose. The ground truth is available as label images and annotation files. Since there are overlaps between windows and other structures, annotation files are better suited as a ground truth. The structure of the given annotation files is proprietary. To make it processable, we converted it into the more suitable PASCAL VOC format. For our evaluation, we used the ICG Graz50 dataset (Riemenschneider et al., 2012) containing 50 facade images and the corresponding semantic segmentation. We chose this dataset due to its strict rectangular ground truth labeling. Thus, we could compare predicted bounding boxes with these label images. Some other common datasets, like eTrims (Korč , Förstner, 2009), do not fit due to perspective distortions. For both training and evaluation, rotation of objects were not considered. Finally, we applied the trained network to the textures of a district's LoD2 CityGML model, cf. Section 5.

resolution	avg. accuracy	avg. precision	avg. IoU
1×1	0.94147	0.89194	0.71244
$1/2 \times 1/2$	0.94371	0.89142	0.72967
$1/4 \times 1/4$	0.93019	0.82679	0.68058
$1/8 \times 1/8$	0.86585	0.66682	0.42009
$1/16 \times 1/16$	0.80901	0.44350	0.10312
$1/32 \times 1/32$	0.81225	0.31363	0.00475

Table 1. Performance of window detection with Faster R-CNN on the ICG Graz50 dataset using different image resolutions.

resolution	avg. accuracy	avg. precision	avg. IoU
1×1	0.98378	0.83387	0.48817
$1/2 \times 1/2$	0.98333	0.87482	0.51574
$1/4 \times 1/4$	0.98277	0.77461	0.52242
$1/8 \times 1/8$	0.97775	0.73321	0.43813
$1/16 \times 1/16$	0.96756	0.47705	0.21227
$1/32 \times 1/32$	0.96829	0.29476	0.02556

Table 2. Performance of door detection with Faster R-CNN on the ICG Graz50 dataset using different image resolutions.

For evaluation of image segmentation, it is standard to count every classified pixel with one of four sums: true positive (tp), false positive (fp), true negative (tn) and false negative (fn). Using these numbers, we are able to calculate different evaluation scores like precision $\mathrm{tp}/(\mathrm{tp}+\mathrm{fp})$, accuracy $(\mathrm{tp}+\mathrm{tn})/(\mathrm{tp}+\mathrm{tn}+\mathrm{fp}+\mathrm{fn})$, and Intersection over Union $\mathrm{IoU} = \mathrm{tp}/(\mathrm{tp}+\mathrm{fp}+\mathrm{fn})$ as the most meaningful evaluation score since it does not take background pixels into account. The average scores in Tables 1, 2 and 3 are means taken over the scores calculated separately for each image in the dataset.

We did evaluation on different resolutions of facade images to check the usability of low resolution data, which is common in already existing LoD2 CityGML models. We reduced the width and height of the image five times by a factor of two. The original resolution of the images in the ICG Graz50 datasets varies between 200 and 500 pixels in height and width.

We noticed that the coarser the resolution the larger the size of bounding boxes becomes, see Figure 3. Astonishingly, one can obtain somewhat better IoU scores with a reduced resolution, see Tables 1 and 2. To understand IoU numbers, one has to know that not all windows are segmented in the ICG Graz50 ground truth. Windows that do not belong to the main body of a facade are labeled as sky. These windows exist in the images and therefore also get correctly detected. Thus, the false positive count is too high. Nevertheless, our average window segmentation accuracy of 0.94 exceeds by far the value of 0.84 of (Rahmani , Mayer, 2018).

4. ALIGNMENT WITH A LINEAR PROGRAM

An elegant approach for aligning window and door rectangles is to solve a mixed integer linear optimization problem (MILP). MILPs have been successfully applied to not only problems of Operations Research but also to several tasks in geometric modeling. For example, optimum binary labelings can be used for

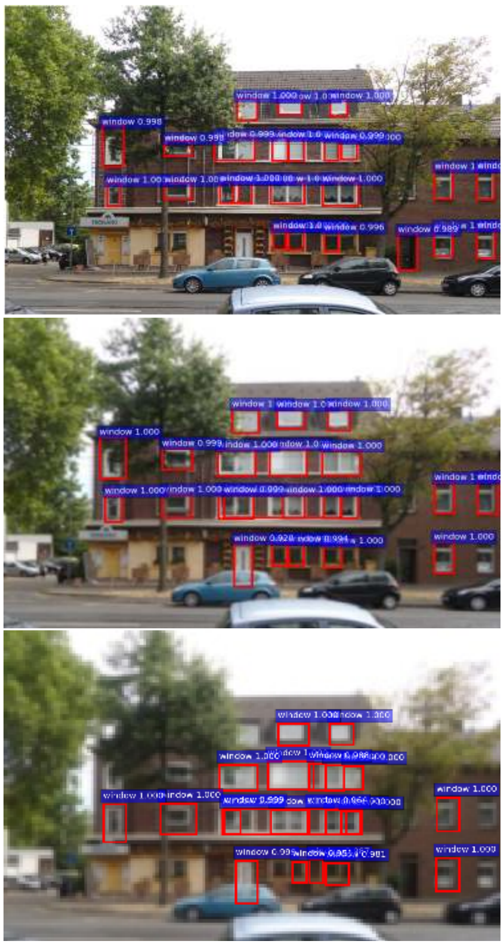

Figure 3. Example for problems in window detection on low resolution images. From top to bottom: original resolution (2590 × 1715 pixel), 1/16 of the original resolution and 1/32 of the original resolution.

surface reconstruction, see (Boulch et al., 2014) and the literature cited there. MILPs can also be used to establish right angles and symmetry in roof structures of CityGML models, see (Goebbels , Pohle-Fröhlich, 2018). But here the approach is completely different. The idea is not to successively cluster windows with same properties but to let the optimization algorithm decide about coordinate changes in one step. All alignment goals, like a maximum number of identical coordinates, few different window sizes and minimal coordinate changes, can be obtained with one MILP. If one adjusts window sizes in a first step and aligns window edges in a second step, then the second step has to be performed on modified and not on original coordinates. This might lead to a suboptimal solution.

4.1 Model Parameters

With N we denote the number of window and door rectangles. Let t_k and b_k be float variables, $k \in [m] := \{1, \ldots, m\}$, that will contain different height positions (y-coordinates) of top and bottom edges of rectangles in an optimal alignment, see Figure 4. A minimum number of these values should be shared by a maximum number of rectangles. Thus, each value represents an alignment class. Let l_k and r_k, $k \in [n]$, be float variables containing different left and right horizontal positions of rectangles in the final arrangement. Float variables h_k, $k \in [m']$, and w_k, $k \in [n']$, are used to store different final

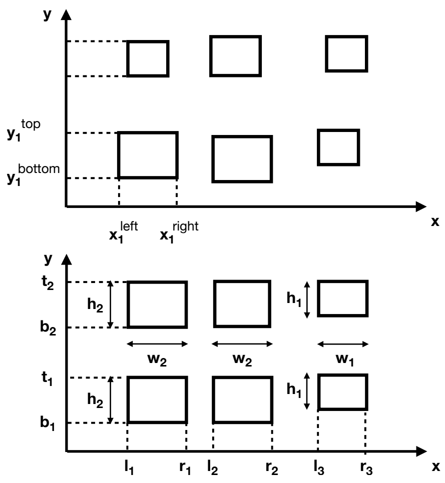

Figure 4. Names of float variables for rectangle coordinates (upper image) and for alignment classes

height and width values. Number m denotes an upper bound for the expected number of different top or bottom window coordinates. Accordingly, number n is an upper bound for window column positions. Obviously, it does not make sense to choose $m, n > N$. But to reduce the number of variables, upper bounds smaller than N are welcome. The estimated number of building floors (height of the facade divided by three) approximately is a lower bound for choosing m. Half the facade's length (in meter) is an estimate of a lower bound for choosing n because typically most left and right coordinates of windows are shared between floors. Facades height and length values are available from the CityGML wall object. However, the algorithm can be applied in connection with facade images independently of CityGML models, for example with ICG Graz50 data that we used to validate the neural network. If an image with r rows and c columns is given, one can assume an isotropic scaling factor s and apply heuristics

$$\frac{s \cdot r}{3} \cdot \frac{s \cdot c}{2} \approx N \iff s \approx \sqrt{\frac{6 \cdot N}{r \cdot c}} \tag{1}$$

to estimate lower bounds $m > \frac{s \cdot r}{3}$ and $n > \frac{s \cdot c}{2}$. For example, one can initialize n and m with these lower bounds times two, cf. Figure 5. This allows many shifted window positions. Of course, m and n can also be determined more precisely based on detected window positions.

In general, there are only few different types of windows and doors. Thus, n' and m' can be chosen as small values. If m, n, m', or n' are chosen too small, the arrangement of windows and doors will only be optimized partially. If they are chosen too large, some alignment classes are empty, i.e., their variable values are not linked with rectangle coordinates. The power of the presented MILP is that it will automatically group coordinates into clusters without a-priori knowledge about the number of clusters. Too many alignment classes do not have impact on the result and only moderate impact on processor time[1]. Figure 5 shows the effect. Parameter settings allow for $m' = n' = 5$ different window height and width values, $n = m = 8$ different classes of x- and y-positions. This leads to a computation time

[1]Processor times were measured on a MacBook Pro with an 2.3 GHz i5 processor and 16 GB RAM. Running times are much faster because processor times are sums of execution times of all parallel threads. We observed running times that equal processor times divided by four.

of 61 s. In this setting, $m = 8$ is chosen too small to align all windows. No class is assigned to one top and one bottom position of windows in the right column. If we set $n = m = 33$ and $m' = n' = 10$, the same solution is computed in 132 s processor time by ignoring most alignment class variables. By setting n and m to values twice the lower bounds obtained with heuristics (1), we obtain $m = 9$ and $n = 14$ (while keeping $m' = n' = 10$). With these parameters, processor time is reduced to 17 s.

Each rectangle i is given via four coordinates x_i^{left}, x_i^{right}, y_i^{top}, and y_i^{bottom}, see Figure 4. We measure positions in pixels of the facade's texture so that we do not have to convert them into meter. For each coordinate, we introduce two non-negative float variables to store changes:

$$\Delta x_i^{\text{left}\,+},\quad \Delta x_i^{\text{left}\,-},\quad \Delta x_i^{\text{right}\,+},\quad \Delta x_i^{\text{right}\,-},$$
$$\Delta y_i^{\text{top}\,+},\quad \Delta y_i^{\text{top}\,-},\quad \Delta y_i^{\text{bottom}\,+},\quad \Delta y_i^{\text{bottom}\,-}.$$

For example, the left coordinate x_i^{left} will be changed to $x_i^{\text{left}} + \Delta x_i^{\text{left}\,+} - \Delta x_i^{\text{left}\,-}$. Instead of measuring changes by taking absolute values or squares in a non-linear fashion, now the linear term $\Delta x_i^{\text{left}\,+} + \Delta x_i^{\text{left}\,-}$ can be used. We do not allow arbitrary large coordinate changes, thus variables $\Delta x_i^{\text{left}\,+}, \dots, \Delta y_i^{\text{bottom}\,-}$ have to be upper bounded by a threshold value ε. We chose ε to be a third of the smallest window height or width measured in pixels. This prohibits degeneration of windows. But of course, an absolute tolerance measured in meters can be used, too. The neural network equips estimated bounding boxes with probabilities. They measure the confidence of matching boxes with real windows or doors. If larger coordinate changes for possibly wrong boxes are allowed then corresponding coordinates will have less impact on coordinate changes of correctly estimated windows. However, all possibilities were very close to one in our tests. Therefore, individual, window-dependent bounds $\varepsilon(i)$, $i \in [N]$ do not promise to give better solutions.

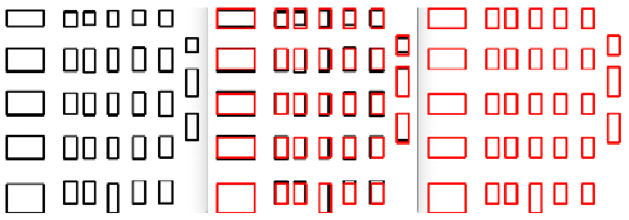

Figure 5. Original and optimized layout of 33 detected windows.

4.2 Integer Linear Program

If binary variable $a_{i,k}^{\text{top}} \in \{0,1\}$ is set to one then rectangle i has an upper y-coordinate $t_k = y_i^{\text{top}} + \Delta y_i^{\text{top}\,+} - \Delta y_i^{\text{top}\,-}$. In the same manner, binary variables $a_{i,k}^{\text{bottom}}$, $a_{i,k}^{\text{left}}$, $a_{i,k}^{\text{right}}$, $a_{i,k}^{\text{height}}$, and $a_{i,k}^{\text{width}}$ indicate if corresponding values match with b_k, l_k, r_k, h_k, and w_k, respectively. To define an objective function, let

$$F := \frac{1}{8N\varepsilon(\max\{n+n', m+m'\}+1)}.$$

We align windows and doors by maximizing the following objective function

$$\sum_{i=1}^{N}\left(\sum_{k=1}^{m}\frac{a_{i,k}^{\text{top}}+a_{i,k}^{\text{bottom}}}{k} + \sum_{k=1}^{m'}\frac{a_{i,k}^{\text{height}}}{m+k}\right.$$
$$\left. + \sum_{k=1}^{n}\frac{a_{i,k}^{\text{left}}+a_{i,k}^{\text{right}}}{k} + \sum_{k=1}^{n'}\frac{a_{i,k}^{\text{width}}}{n+k}\right) \tag{2}$$
$$-F\sum_{i=1}^{N}\left[\Delta x_i^{\text{left}\,+} + \Delta x_i^{\text{left}\,-} + \Delta x_i^{\text{right}\,+} + \Delta x_i^{\text{right}\,-}\right.$$
$$\left. + \Delta y_i^{\text{top}\,+} + \Delta y_i^{\text{top}\,-} + \Delta y_i^{\text{bottom}\,+} + \Delta y_i^{\text{bottom}\,-}\right].$$

Our aim is to maximize the number of alignments. At the same time, the sum of all coordinate changes should be minimal as a secondary goal, see Figure 5. Factor F in (2) ensures that an additional alignment gains more profit than it might reduce due to all coordinate changes. Also, a minimum number of different alignment classes should be used. We enforce this by weighting a class of index k with weight $\frac{1}{k}$, $\frac{1}{m+k}$, or $\frac{1}{n+k}$. This prohibits a class from being split into two or more because the value of the objective function would decrease. We use smaller weights $\frac{1}{m+k}$ and $\frac{1}{n+k}$ for width and height classes. Thus, aligning to rows and columns becomes a higher priority than using same window sizes. If one uses weights $\frac{1}{k}$ instead of $\frac{1}{m+k}$ and $\frac{1}{n+k}$ in (2), alignment to a height (or width) value might be weighted higher than alignment to a top or bottom class. This results in the artifact marked in Figure 6.

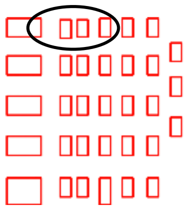

Figure 6. If height or width alignment takes precedence over top or bottom alignment, artifacts occur, cf. Figure 5.

A maximum of (2) has to be found subject to following restrictions: All window attributes have to be element of at most one alignment class, i.e., for all $i \in [N]$

$$\sum_{k=1}^{m} a_{i,k}^{\text{top}} \le 1,\quad \sum_{k=1}^{m} a_{i,k}^{\text{bottom}} \le 1,\quad \sum_{k=1}^{m'} a_{i,k}^{\text{height}} \le 1,$$
$$\sum_{k=1}^{n} a_{i,k}^{\text{left}} \le 1,\quad \sum_{k=1}^{n} a_{i,k}^{\text{right}} \le 1,\quad \sum_{k=1}^{n'} a_{i,k}^{\text{width}} \le 1.$$

If a binary variable is set to one, the modified coordinate or size has to fit with the class value, for example ($i \in [N]$, $k \in [m]$)

$$-M[1-a_{i,k}^{\text{top}}] \le y_i^{\text{top}} + \Delta y_i^{\text{top}\,+} - \Delta y_i^{\text{top}\,-} - t_k \le M[1-a_{i,k}^{\text{top}}].$$

The constant M has to be chosen sufficiently large. We set M to be the maximum number of rows and columns of the facades image. Thus, the condition is no limitation at all in case of $a_{i,k}^{\text{top}} = 0$. But if $a_{i,k}^{\text{top}} = 1$ then the changed top coordinate has to equal t_k. This is a standard trick in linear programming. Binary variables are used for all other corresponding restrictions. For example, we enforce height conditions with ($i \in [N]$, $k \in [m']$)

$$\begin{aligned} -M[1-a_{i,k}^{\text{height}}] &\le y_i^{\text{top}} + \Delta y_i^{\text{top}\,+} - \Delta y_i^{\text{top}\,-} \\ &\quad - y_i^{\text{bottom}} - \Delta y_i^{\text{bottom}\,+} + \Delta y_i^{\text{bottom}\,-} - h_k \\ &\le \quad M[1-a_{i,k}^{\text{height}}]. \end{aligned}$$

So far, the MILP does not check if rectangles overlap. Unfortunately, the neural network might deliver overlapping bounding boxes, and the described optimization procedure also might generate overlaps. Thus, we add restrictions to resolve these problems if overlaps are within the tolerance 2ε. Otherwise, we merge overlapping windows or doors to become a single rectangle. For two rectangles $r_1 \neq r_2 \in [N]$ let $i \neq j \in \{r_1, r_2\}$ such that $y_i^{\text{bottom}} \leq y_j^{\text{bottom}}$, and let $k \neq l \in \{r_1, r_2\}$ such that $x_k^{\text{left}} \leq x_l^{\text{left}}$. Then optimized rectangles r_1 and r_2 might overlap if $y_i^{\text{top}} \geq y_j^{\text{bottom}} - 2\varepsilon$ and $x_k^{\text{right}} \geq x_l^{\text{left}} - 2\varepsilon$. In such a situation, we add an additional restriction either for the x- or y-direction to avoid overlapping. If $y_i^{\text{top}} - y_j^{\text{bottom}} < x_k^{\text{right}} - x_l^{\text{left}}$, we add condition

$$\begin{aligned} & y_i^{\text{top}} + \Delta y_i^{\text{top}+} - \Delta y_i^{\text{top}-} \\ \leq \quad & y_j^{\text{bottom}} + \Delta y_j^{\text{bottom}+} - \Delta y_j^{\text{bottom}-} - \varepsilon_0, \end{aligned}$$

otherwise, for a small minimum pixel distance $\varepsilon_0 > 0$, we use

$$x_k^{\text{right}} + \Delta x_k^{\text{right}+} - \Delta x_k^{\text{right}-} \leq x_l^{\text{left}} + \Delta x_l^{\text{left}+} - \Delta x_l^{\text{left}-} - \varepsilon_0.$$

Distance $\varepsilon_0 > 0$ should exist between windows or doors.

After maximizing object function (2) subject to these restrictions, coordinates of each bounding rectangle are updated, for example y_i^{top} is replaced by $y_i^{\text{top}} + \Delta y_i^{\text{top}+} - \Delta y_i^{\text{top}-}$.

4.3 Performance Considerations

In general, MILPs are NP hard. This algorithm's processor time mostly depends on the number N of windows and doors and on the solver to execute the MILP but also on the tolerance ε. When using the Callable Library of IBM's state of the art commercial optimizer ILOG CPLEX Optimization Studio 12.8[2] with standard parameters, the longest observed processing time for aligning a facade of our test CityGML dataset was less than 45 s, see Figure 7. The average number of windows plus doors is 6.96, the average number of modified windows and doors is 6.44, and the average processor time of CPLEX is 1.43 s whereas the median is 0.2 s. Due to parallel execution, running times are much faster.

However, optimization of detected window rectangles of larger ICG Graz50 facades was much more time-consuming, see upper right box plot in Figure 8. It shows processor times for 49 out of 50 facades. There were three outliers: One facade exceeded a processor time limit of ten hours (excluded from box plot), one was optimized in 5.6 h and one in 4.2 h processor time. Nevertheless, the MILP formulation of the aligning problem is applicable for practical purposes in connection with a time limit. We reduced processor times by adding additional

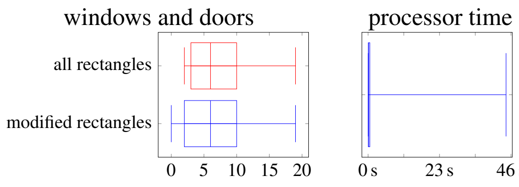

Figure 7. Box plots with data of 160 optimized facades.

problem dependent constraints. Due to the objective function, the number of windows that are aligned to the value of t_1 is

[2]see https://www.ibm.com/customer-engagement/commerce

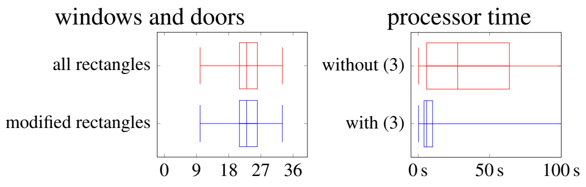

Figure 8. Alignment of windows that were detected on the 50 facade images of the ICG Graz50 collection.

	avg. accuracy	avg. precision	avg. IoU
without MILP	0.94147	0.89194	0.71244
with MILP	0.93958	0.88615	0.70318

Table 3. Detection results of Faster R-CNN without and with MILP post-processing.

the largest. Then the numbers decrease with t_2, t_3, and so on. However, we can also sort values of all class variables in ascending order, e.g. by requiring $t_1 < t_2 < t_3 < \cdots < t_m$. Thus, because coordinate changes are limited by ε, we can mutually exclude binary variables from being one. For example, let indices i and j be chosen such that $y_i^{\text{top}} > y_j^{\text{top}} + 2\varepsilon$. Then

$$\sum_{s=1}^{k} a_{i,s}^{\text{top}} + a_{j,k}^{\text{top}} \leq 1 \tag{3}$$

holds true. If we add such conditions to the linear program for all six groups of binary variables (that indicate membership to top, bottom, left, right, height, and width alignment classes), then we can indeed shorten optimization times. The lower right box plot in Figure 8 shows the increase in performance for ICG Graz50 data. By using the additional constraints (3), the median processing time could be reduced from 27.59 s to 5.97 s. Without (3), one computation exceeded a time limit of ten hours. With constraints (3), the longest processor time was 1676 s, and all 50 facades could be processed.

The goal of the MILP is to reduce noise and to adjust facades according to the rules that often windows are of equal size and that they are aligned to each other. The aim is not to increase detection results. However, accuracy, precision and IoU do not change significantly if the MILP is applied on detected windows and doors, see Table 3. Thus, beautification does not lead to worse layouts.

5. ADDING WINDOWS AND DOORS TO LOD2 CITYGML MODELS

We apply the neural network to wall textures in LoD2 CityGML 2.0.0 models. Using vertex coordinates of wall polygons in connection with their given texture coordinates, images are corrected in perspective and clipped at walls' boundaries. Resulting images are scaled to fit with the neural network's input layer. This leads to transformed texture coordinates. For alignment, the linear program then is performed on such scaled data, too. To map results back to wall polygons, we utilize the transformed texture coordinates. For each rectangle, our tool cuts a corresponding opening into the counter-clockwise oriented wall polygon by adding a clockwise oriented interior polygon ("gml:LinearRing"). Thus, we now use LoD3 instead of LoD2 CityGML structures. The tool also adds an opening object of class "Window" or "Door" for each rectangle to the corresponding "WallSurface" object, see (Gröger et al., 2012, p. 91) and Figures 9 and 10. Such opening objects can be used to model windows and doors in detail. At this point, we just define a window or door by a simple counter-clockwise oriented rectangle. However, with nearly no effort, one could replace the rectangle by a prototype 3D window or door model using implicit geometries. After adding openings to wall surfaces, the tool has to update texture coordinates in "ParameterizedTexture" objects by adding a new "textureCoordinates" object for each new interior polygon. It is linked to the polygon via the polygon's "LinearRing" identifier, see (Gröger et al., 2012, pp. 38–42).

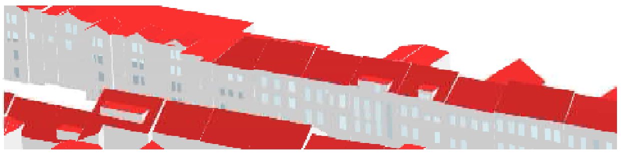

Figure 9. LoD3 building models with windows, visualized with Azul CityGML viewer.

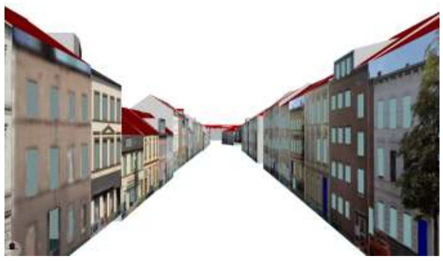

Figure 10. LoD3 building models with windows, doors and textures, visualized with FZKViewer.

6. CONCLUSION AND FUTURE WORK

We have shown that Faster R-CNN can be used to segment windows and doors, even on low resolution facades. The focus of this paper is on the improvement of LoD2 CityGML models using object detection, but we did additional tests using a modified Faster R-CNN which is capable of predicting depth information for more detailed facade modeling. To this end, we enhanced the final step in the Faster R-CNN pipeline, the R-CNN. The original R-CNN has the task to classify region proposals and to adjust bounding boxes according to predicted classes. To estimate a depth value, we integrated an FC layer with 256 units additionally to the two existing FC layers responsible for class prediction and bounding box regression, see Figure 11. Similar to class prediction, each depth value has its probability score. We defined the depth with a range from 0 to 255, with the facade wall placed at 128. Thus, it can be used for detailed modeling of windows but also of other objects like balconies. Unfortunately, we could not find annotated openly available datasets that would serve as ground truth for training this depth value estimation. To be able to test our concept of additional depth prediction for facade objects, we made simple assumptions for depth values. For example, a window is more likely to be placed behind the wall of the facade. According to this, we added fixed depth values for our two classes to the ground truth. Additionally, we tested openly available neural networks for generating ground truth depth. Since results were fuzzy and not usable, more research is needed to find better performing neural network architectures.

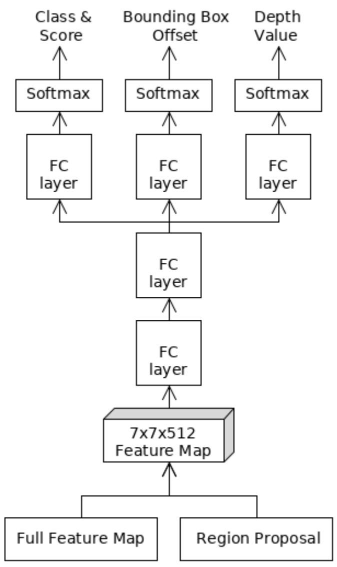

Figure 11. Enhanced R-CNN architecture

Furthermore, our current procedure does only distinguish between doors and windows but does not determine types of windows. Based on a catalog of parameterized standard window types, one could train the neural network to directly propose a 3D window model that can be used as an opening object. Apart from that, image content of detected bounding boxes could be matched with the catalog in a second step. Both approaches could be an alternative to the depth predicting approach mentioned before and would additionally allow dealing with non-rectangular window frames.

ACKNOWLEDGEMENTS

This work was supported by a generous hardware grant from NVIDIA. We also used an academic license of IBM's CPLEX at no cost.

REFERENCES

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., Çöltekin, A., 2015. Applications of 3D city models: State of the art review. *ISPRS Int. J. Geo-Inf.*, 4, 2842–2889.

Boulch, A., de La Gorce, M., Marlet, R., 2014. Piecewise-planar 3D reconstruction with edge and corner regularization. *Computer Graphics Forum*, 33, 55–64.

Finlayson, Graham D., Hordley, Steven D., Drew, Mark S., 2002. Removing shadows from images. *Proc. 7th European Conference on Computer Vision-Part IV*, ECCV '02, Springer, Berlin, 823–836.

Gadde, R., Jampani, V., Marlet, R., Gehler, P. V., 2018. Efficient 2D and 3D facade segmentation using auto-context. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40, 1273-1280.

Girshick, Ross, 2015. Fast R-CNN. *Proceedings of the IEEE international conference on computer vision*, 1440–1448.

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, Malik, Jitendra, 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, IEEE Computer Society, Washington, DC, USA, 580–587.

Goebbels, S., Pohle-Fröhlich, R., 2018. Beautification of city models based on mixed integer linear programming. *Proceedings OR 2018*, Springer, Berlin, in print.

Gröger, G., Kolbe, T. H., Nagel, C., Häfele, K. H. 2012. *OpenGIS City Geography Markup Language (CityGML) Encoding Standard. Version 2.0.0.* Open Geospatial Consortium.

Guo, Ruiqi, Dai, Qieyun, Hoiem, D., 2011. Single-image shadow detection and removal using paired regions. *Proc. 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, IEEE, Washington, DC, 2033–2040.

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2980–2988.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Korč, F., Förstner, W., 2009. eTRIMS image database for interpreting images of man-made scenes. *Technical Report, Dept. of Photogrammetry, University of Bonn*, 1–12.

Kulkarni, V., Nagesh, R., Wu, H, 2011. Window detection in frontal facades. *Technical Report, University of California, Berkeley, CA*, 1–7.

Lee, S. C., Nevatia, R., 2004. Extraction and integration of window in a 3D building model from ground view images. *Proc. 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, II–113–II–120.

Li, W., Yang, M. Y., 2016. Efficient semantic segmentation of man-made scenes using fully-connected conditional random field. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLI-B3, 633–640.

Liu, Hantang, Zhang, Jialiang, Zhu, Jianke, Hoi, Steven C. H., 2017. Deepfacade: A deep learning approach to facade parsing. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2301–2307.

Long, Jonathan, Shelhamer, Evan, Darrell, Trevor, 2015. Fully convolutional networks for semantic segmentation. *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3431–3440.

Malihi, Shirin, Valadan Zoej, Mohammad Javad, Hahn, Michael, Mokhtarzade, Mehdi, 2018. Window detection from UAS-derived photogrammetric point cloud employing density-based filtering and perceptual organization. *Remote Sensing*, 10, 2072-4292.

Martinović, A., Knopp, J., Riemenschneider, H., Gool, L. Van, 2015. 3D all the way: Semantic segmentation of urban scenes from start to end in 3D. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4456–4465.

Mathias, Markus, Martinović, Anđelo, Van Gool, Luc, 2016. ATLAS: A three-layered approach to facade parsing. *International Journal of Computer Vision*, 118, 22–48.

Miljanovic, M., Eiter, T:, Egly, U., 2012. Detection of windows in facades using image processing algorithms. *Indian Journal of Computer Science and Engineering*, 3, 539–547.

Qi, Charles Ruizhongtai, Yi, Li, Su, Hao, Guibas, Leonidas J, 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds), *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., 5099–5108.

Rahmani, K., Huang, H., Mayer, H., 2017. Facade segmentation with a structured random forest. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1/W1, 175–181.

Rahmani, K., Mayer, H., 2018. High quality facade segmentation based on structured random forest, region proposal network and rectangular fitting. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2, 223–230.

Ren, Shaoqing, He, Kaiming, Girshick, Ross, Sun, Jian, 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (eds), *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., 91–99.

Riemenschneider, H., Krispel, U., Thaller, W., Donoser, M., Havemann, S., Fellner, D., Bischof, H., 2012. Irregular lattices for complex shape grammar facade parsing. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 1640–1647.

Simonyan, Karen, Zisserman, Andrew, 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Su, Hang, Jampani, Varun, Sun, Deqing, Maji, Subhransu, Kalogerakis, Evangelos, Yang, Ming-Hsuan, Kautz, Jan, 2018. Splatnet: Sparse lattice networks for point cloud processing. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2530–2539.

Tuttas, S., Stilla, U., 2013. Reconstruction of façades in point clouds from multi aspect oblique ALS. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 91-96.

Tyleček, Radim, Šára, Radim, 2013. Spatial pattern templates for recognition of objects with regular structure. J. Weickert, M. Hein, B. Schiele (eds), *Pattern Recognition*, Springer Berlin Heidelberg, 364–374.

Wan, Guowei, Sharf, Andrei, 2012. Grammar-based 3D facade segmentation and reconstruction. *Computers & Graphics*, 36, 216–223.

Wu, Zhirong, Song, S., Khosla, A., Yu, Fisher, Zhang, Linguang, Tang, Xiaoou, Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1912–1920.

Xingzi Zhang, Franziska Lippoldt, Kan Chen, Johan, Henry, Erdt, Marius, 2019. A data-driven approach for adding facade details to textured LoD2 CityGML models. *Proceedings GRAPP 2019*, 294–301.