Facade Layout Completion with Long Short-term Memory Networks

Simon Hensel
Institute for Pattern Recognition
Niederrhein University of Applied Sciences
Reinarzstrasse 49, Krefeld, Germany
simon.hensel@hs-niederrhein.de

Steffen Goebbels
Institute for Pattern Recognition
Niederrhein University of Applied Sciences
Reinarzstrasse 49, Krefeld, Germany
steffen.goebbels@hs-niederrhein.de

Martin Kada
Institute of Geodesy and Geoinformation Science
Technische Universität Berlin
Kaiserin-Augusta-Allee 104-106, Berlin, Germany
martin.kada@tu-berlin.de

February 6, 2023

Abstract

In a workflow creating 3D city models, facades of buildings can be reconstructed from oblique aerial images for which the extrinsic and intrinsic parameters are known. If the wall planes have already been determined, e.g., based on airborne laser scanning point clouds, facade textures can be computed by applying a perspective transform. These images given, doors and windows can be detected and then added to the 3D model. In this study, the “Scaled YOLOv4” neural network is applied to detect facade objects. However, due to occlusions and artifacts from perspective correction, in general not all windows and doors are detected. This leads to the necessity of automatically continuing the pattern of facade objects into occluded or distorted areas. To this end, we propose a new approach based on recurrent neural networks. In addition to applying the Multi-Dimensional Long Short-term Memory network and the Quasi Recurrent Neural Network, we also use a novel architecture, the Rotated Multi-Dimensional Long Short-term Memory network. This architecture combines four two-dimensional Multi-Dimensional Long Short-term Memory networks on rotated images. Independent of the 3D city model workflow, the three networks were additionally tested on the Graz50 dataset for which the Rotated Multi-Dimensional Long Short-term Memory network delivered better results than the other two networks. The facade texture regions, in which windows and doors are added to the set of initially detected facade objects, are likely to be occluded or distorted. Before equipping 3D models with these textures, inpainting should be applied to these regions which then serve as automatically obtained inpainting masks.

Keywords Deep Learning · LSTM · Facade Reconstruction · Structure Completion · Image Inpainting.

1 INTRODUCTION

Most current 3D city models were created based on airborne laser scanning point clouds. Whereas roofs are clearly visible, the point density of vertical walls is much lower, and walls might be occluded by roofs. Thus, facades are often represented as planar polygons without any details. The position and size of facade objects like windows and doors can be better obtained from oblique aerial images. By utilizing camera parameters, segments from these images can be mapped to become textures of facade polygons. These are low resolution images that might be distorted by

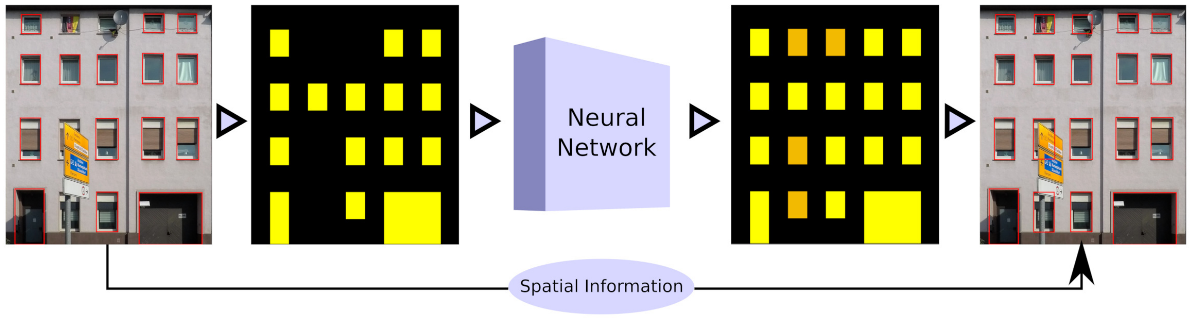


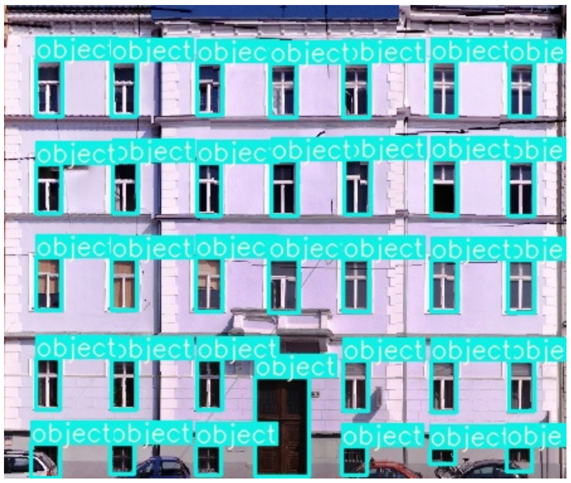
Figure 1: Workflow for facade structure completion: Due to a flag, a satellite dish and a signpost, four windows could not be detected. They were inserted using an LSTM. [Hensel et al., 2021]

occlusions and shadows. Since real facades are not exactly planar (e.g., due to balconies), artifacts from the perspective correction occur as well. However, windows and doors have to be detected on these distorted images in order to obtain semantic 3D facade models. For example, neural networks for object detection and instance segmentation, see Section 2, search for individual object instances but do not consider higher-level layout patterns. This also holds true for the state-of-the-art object detection network "Scaled YOLOv4" [Wang et al., 2021] that is applied in this study. Thus, relationships between windows are not taken into account to also find occluded object instances. But model knowledge can be used to add missing instances in a post-processing step. One way to do this is by applying split grammars that were introduced in [Wonka et al., 2003] for building reconstruction. Such grammars consist of rules that describe the placement and orientation of objects. They allow filling in missing facade objects and also generating facades procedurally from scratch. However, the grammar rules must be appropriate for a particular building style. Individual facade layouts may require individual, manually created rules. Since it is difficult to find a suitable collection of rules, machine learning is an appropriate tool.

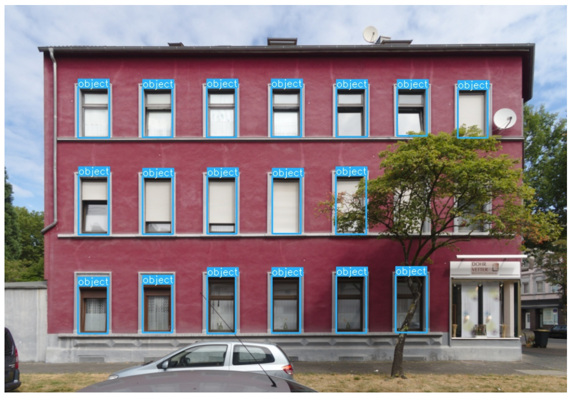
We propose the use of Recurrent Neural Networks (RNNs) with Long Short-term Memory (LSTM) to recover positions and sizes of missing windows and doors within a workflow shown in Figure 1. These networks are commonly applied to time-dependent, one-dimensional input data, e.g., in the context of speech recognition. This leads to an input consisting of incomplete detection results, e.g., from the Scaled YOLOv4 network. These results are represented by (two-dimensional) bounding boxes. We obtain an irregular rectangular lattice (IRL) by extending the edges of the bounding boxes to straight lines. Thus, an IRL is a collection of horizontal and vertical lines. It is called "irregular" because the distances between the lines may vary. Bounding boxes of detected objects are not perfectly aligned, so we simplify the IRL depending on the image resolution. Lines are merged if they are closer than a resolution-specific threshold distance (see Section 3.3). The cells of the IRL are the rectangles that are bounded but not intersected by lines. In the beginning, cells are labeled with the initial detection results. They are either classified as background or as part of a facade object (window/door). The task of the LSTM is to correct the initial labels of the cells so that missing objects are added. To this end, we compare results of the proposed Rotated Multi-Dimensional Long Short-term Memory network (RMD LSTM [Hensel et al., 2021], see Section 4) with results of the Multi-Dimensional Long Short-term Memory (MD LSTM) [Graves et al., 2007] and the Quasi Recurrent Neural Network (QRNN), see [Bradbury et al., 2016], in Section 6.

We are not only interested in the positions and sizes of windows and doors to create semantic 3D building models but also want to texture the models with the given oblique aerial images. Initially, undetected objects that could be added by the RNN approach help to identify occluded or distorted image regions. If a window or door is not detected in the distorted image, we assume that the object and its surrounding is occluded. Thus, an occlusion mask can be computed automatically as the union of all corresponding surroundings. Then, an inpainting algorithm can be applied to fill these regions. Inpainting is a standard computer vision technique that interpolates or extrapolates visual information to fill image regions that are typically defined by a binary masks such that edges and texture patterns should somehow fit with the given information, see, e.g., the overview paper [Mehra et al., 2020]. Our approach to computing occlusion masks based on added objects differs from a more common technique based on segmenting objects that may occlude facades, see e.g. [Chen et al., 2018]. However, it is difficult to segment all objects that cause occlusions because a facade can be occluded by many different objects belonging to many object classes. This difficulty does not occur with our proposed method.

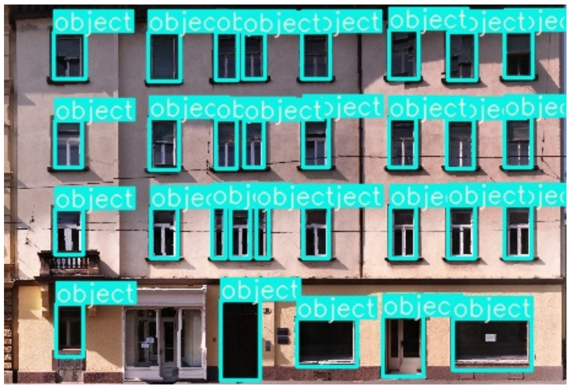
This work builds on the foundation established in [Hensel et al., 2021]. We have rewritten and extended individual sections. We also introduce a new use case in Section 6.2 where the proposed workflow is inserted into a facade inpainting process.



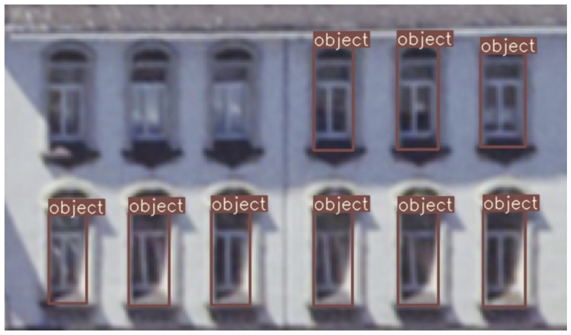
(a)



(b)



(c)



(d)

Figure 2: Examples of object detection using Scaled YOLOv4. (a) and (c) are images from the Graz50 dataset [Riemenschneider et al., 2012], (b) has a high resolution whereas the resolution of image (d) is low.

2 RELATED WORK

We are concerned with a workflow that generates semantic 3D building models. Such models are represented and exchanged with the open data model format CityGML see [Gröger et al., 2007] or more recently with CityJSON[1].

In the presented workflow, windows and doors have to be recognized in facade images. Convolutional Neural Networks (CNNs) have become a standard tool for image-based object detection and segmentation. ResNET [He et al., 2016] was a milestone in Deep Learning. It eliminated the well-known vanishing gradient problem that can occur when training very deep neural networks. Mask R-CNN [He et al., 2017] is an important network architecture for segmenting individual object instances. To this end, it applies an attention mechanism that consists of detecting object instances first. Then it performs segmentation within the bounding boxes of the detection results. In the RetinaNet [Lin et al., 2017], the concept of focal loss is implemented to distinguish between foreground and background. Not only when computational resources are limited, YOLO [Redmon et al., 2016] is a commonly used network. The Scaled YOLOv4 network [Wang et al., 2021] provides differently scaled deep learning models, which are chosen according to a balance of image size, channels and layers. On the MC COCO dataset, the network reached an average

[1]https://cityjson.org/specs/ (accessed: February 6, 2023)

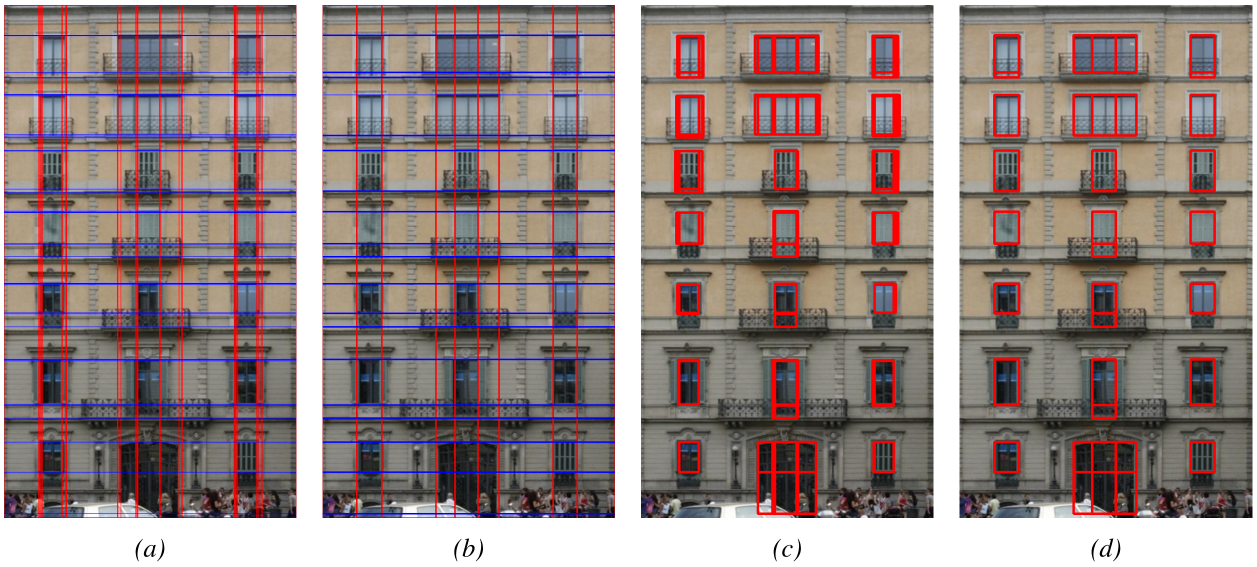


Figure 3: Images *(a)* and *(b)* show an IRL representation of an example facade from the CMP dataset. Images *(c)* and *(d)* show the corresponding bounding boxes. IRL in *(b)* was computed by merging lines of the IRL in *(a)*. [Hensel et al., 2021]

precision of 64.8% and outperformed the EfficientDet neural network [Tan et al., 2020] with the former best average precision of 53.0%.

We use bounding boxes of detected objects to generate an IRL. Unfortunately, the boxes may not be properly aligned. However, they often can be aligned by slightly shifting their position and changing their size. This can be done by solving a combinatorial optimization problem. Such a mixed integer linear program with integer and float variables is presented in [Hensel et al., 2019]. By using predefined bounding box positions, the float variables can be eliminated so that a faster integer linear program can be applied, see [Hu et al., 2020].

If occluded facade regions are known (e.g., by applying semantic segmentation of trees), image inpainting techniques can be applied to fill these regions prior to object detection. Vice versa, we use image inpainting with masks obtained from analyzing object detection results.

Diffusion based inpainting methods fill regions by propagating information from the boundary to the interior. Typically, they are applied to deal with small regions. If larger regions have to be filled, texture synthesis methods like example-based inpainting are used.

General inpainting algorithms have been adopted to the completion of facade images. In [Dai et al., 2013], a Random Forest-based method is used to obtain a semantic segmentation of facade elements. The edges of the segment boundaries are used to create an IRL. The corresponding cells are initially labeled based on the semantic segmentation, similarly to our approach based on bounding boxes, see Section 3.3. The algorithm performs example-based inpainting by copying cell content to cells that have to be filled. To this end, the IRL is interpreted as an undirected graph and a graph labeling problem is solved to ensure structural consistency.

The article [Huang et al., 2014] deals with another inpainting algorithm that is applied in the context of facade reconstruction. The algorithm detects line segments in an image showing edges. It then clusters line segments according to vanishing points of the corresponding lines. Image regions that are covered by line segments belonging to two different vanishing points are interpreted as 2D representations of a 3D plane. This information is then used to continue textures in connection with various cost functions that measure appearance, guidance, orthogonal direction, and proximity.

With the advance of Generative Adversarial Networks (GANs), see [Goodfellow et al., 2014], deep learning based inpainting techniques like the application of the Wasserstein GAN [Arjovsky et al., 2017] in [Yu et al., 2018] have been developed. The EdgeConnect network [Nazeri et al., 2019] applies a GAN to edge images. Instead of using partial convolutions that are restricted to non-occluded pixels, DeepFillv2 [Yu et al., 2019] learns how to apply a convolution mask that is also learned. This mechanism is called gated convolution.

Batch normalization is a standard technique in neural network training. With regard to inpainting, features of occluded regions can lead to mean and variance shifts. Instead of batch normalization, a region normalization method to overcome this problem is proposed in [Yu et al., 2020]. Given an inpainting mask, the image is divided into multiple regions. Instead of performing normalization with the images in a batch, now features are normalized with respect to the image regions.

For the application of general purpose inpainting algorithms, it is necessary to specify the regions that have to be filled. However, an automated 3D building reconstruction workflow requires automatically derived masks for facade textures. In [Chen et al., 2020], the mask is derived from a 3D-to-2D conversion of a facade point cloud. Regions without points define components of the mask. Subsequent to the definition of the mask, inpainting is performed with a GAN.

Model knowledge is applied in [Kottler et al., 2020] where object detection results are used to refine a mask obtained from a segmentation network. Based on object detection, patches are marked that have to be preserved by the inpainting algorithm. The patches define important facade structures.

Our proposed workflow inter- and extrapolates a pattern of windows and doors. Thus, it has to know about typical facade layouts. Such layouts can be either learned (as proposed here) or explicitly given in terms of formal grammars like split grammars introduced in [Wonka et al., 2003].

In [Teboul et al., 2011], shape grammars are used to improve facade object segmentation. The terminal symbols of a grammar correspond with facade object classes. An initial probability map for terminal symbols is gained from a discriminatively trained classifier. In an iterative process, segmentation labels are optimized by applying grammar-based constrains. During this process an immediate and cumulative reward system is used for segmentations of windows, doors, and other facade objects which should better represent reality.

Formal grammars can be also applied to other data than 2D images. For example, in [Dehbi et al., 2016] point cloud data is analyzed by counting the number of points along horizontal and vertical lines with a kernel density estimation. Openings like windows and doors are characterized by lower point counts. Thus, these data can be interpreted with a weighted attributed context-free grammar to refine a facade model. The attributes define semantic rules which are additionally weighted with probabilities.

Since it is difficult to cover all architectural styles with explicitly defined grammars, we apply deep learning with recurrent neural networks. The concept of RNN and LSTM architectures is to equip a neural network with memory and access to previous predictions, cf. [Sherstinsky, 2020]. Such networks are often applied to time-dependent, one-dimensional data. For example, they are used for speech recognition and text understanding, cf. [Salehinejad et al., 2017] and [Mtibaa et al., 2020]. With regard to such applications, they can outperform other network architectures, see [Zhang and Wang, 2016]. Extensions of LSTMs are the MD LSTM [Graves et al., 2007] and the QRNN [Bradbury et al., 2016]. The proposed RMD LSTM is built upon the MD LSTM, cf. [Hensel et al., 2021]. To our knowledge, LSTMs have not been applied to the problem of facade reconstruction by other research groups so far. Due to memory being passed through multiple neurons and a serialized input of two-dimensional data, RNN and LSTM architectures have a high demand for hardware resources. Thus, the size of the input data has to be reduced, see Section 3.3.

3 OBJECT DETECTION AND DATA PREPARATION

This section explains how we prepared data from different data sources and used it for training and testing. While training data was generated randomly, testing data also contained facade layouts based on object detections computed with Scaled YOLOv4.

3.1 Random Data Generation

Both the CMP dataset [Tyleček and Šára, 2013] and the Graz50 dataset [Riemenschneider et al., 2012] contain facade images and annotations for facade object classes. We restrict ourselves to windows and doors that are represented by rectangular regions taken from the datasets. To train and test the networks, window and door rectangles were removed randomly so that the networks had to add them again in their predictions. Hence, the networks did not see RGB images but only facade layouts consisting of window and door rectangles. This information was coded within an IRL, see Section 1.

The CMP dataset consists of 606 facades whereas the Graz50 dataset provides 50 images with corresponding annotations for segmentation. The architectural style of CMP facades differs from the style of Graz50 facades. We utilized this to avoid overfitting as we trained only with the larger CMP dataset but tested with the Graz50 facade layouts.

This also showed that the networks are able to complete facade layouts of arbitrary facade types. One could either generate static test and training data prior to applying the networks or one could generate input dynamically during training and testing by randomly removing facade objects. We apply both methods. To avoid storing huge numbers of facade layouts, we applied the latter technique to dynamically remove 20% of windows and doors within the training iterations. This can additionally help to avoid overfitting, cf. [Tobin et al., 2017]. Since it is easier to compare results on static data, we tested on fixed data that was also generated by removing 20% of facade objects.

3.2 Object Detection

In addition to training and testing with data generated from the CMP and Graz50 datasets by randomly removing facade objects, we also performed tests with results of real object detection. For this purpose, we used the Scaled YOLOv4 network [Wang et al., 2021], which we already mentioned in Section 1. For our purposes, we trained the Scaled YOLOv4 network on the CMP facade dataset to detect windows and doors. We only distinguished between objects and non-objects (binary detection). Subsequently, this trained network was used to generate additional test data. Figure 2 shows results for two Graz50 images, a manually taken photo and a facade derived from oblique aerial imaging.

3.3 Data Preparation

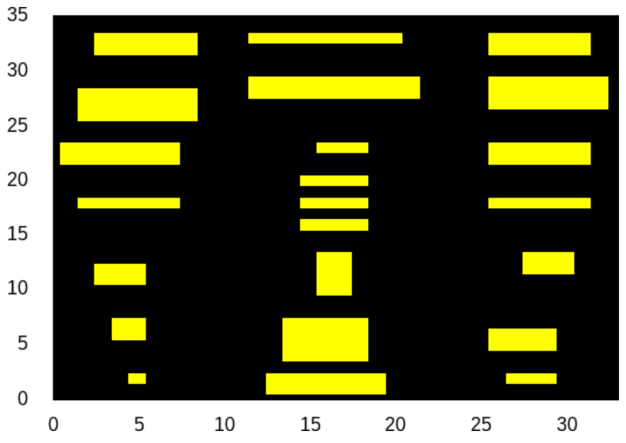
Instead of directly using rectangles representing doors and windows as input for the LSTM networks, a matrix is generated based on an IRL. An initial IRL is obtained from the straight lines which are defined by extending the edges of the rectangles, cf. Section 1. The cells of this lattice are labeled to represent background (label value 0) or windows and doors (label value 1), and the label values become entries of a matrix M. For example, the facade in Figure 1 is represented by the matrix in Formula (1).

$$M = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&1&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&1&0&1&0&1&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&1&0&1&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&1&0&1&1&1&0\\ 0&1&0&0&0&0&0&1&1&1&0 \end{bmatrix}. \tag{1}$$

Since detected facade objects are often not properly aligned, the number of cells and thus the size of the matrix is significantly greater than necessary and might exceed an acceptable input size of the RNN networks. Therefore, we reduce the number of lattice lines of the initial IRL and the size of the matrix by merging neighboring lines. To this end, we iteratively determine neighboring lines of the same orientation within a threshold distance of eight pixels. These lines are replaced with a single line by computing a mean position. Figure 3 (b) shows how an IRL and its matrix change by merging close lines in comparison to Figure 3 (a). Figures 3 (c) and 3 (d) show the corresponding bounding boxes whereas Figure 4 visualizes the corresponding matrix representations. It can be seen that the facade layout does not change significantly but also that merging lines can lead to patterns and symmetry. Alternatively to the chosen merging approach, a pre-processing step could be applied that aligns rectangles by solving an optimization problem, see [Hensel et al., 2019] or [Hu et al., 2020], cf. Section 2.

Whereas the RNN networks work on matrix representations, the spatial coordinates of rectangles representing windows and doors are required to generate 3D models based on the network output. Therefore, the simplified IRL is stored in addition to the matrix representation.

For the considered datasets, matrices had sizes between 10 and 100 rows and columns. Due to hardware restrictions, we choose a maximum network input size of 25×25 labels. Smaller matrices were fitted using zero padding. We experimented with scaling to also fit larger matrices, but that did not lead to acceptable results. Therefore, we only worked with facades that result in matrices with at most 25 rows and columns. This leaves 359 facades of the CMP and 46 facades of the Graz50 data for training and testing. Whereas the MD LSTM and the RMD LSTM networks are able to process 2D matrix input, matrices have to be serialized to be processed by the QRNN.


(a) Matrix representation of IRL before merging of lines within threshold distance.

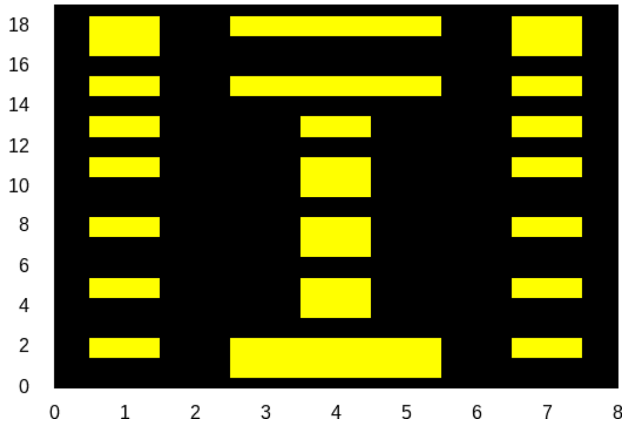

(b) Resulting Matrix with merged horizontal and vertical lines.

Figure 4: Impact of merging vertical and horizontal lattice lines in the IRL of Figure 3*(a)*. Note that axes are of a different scale, such that the resulting matrix of the simplified IRL is much smaller. [Hensel et al., 2021]

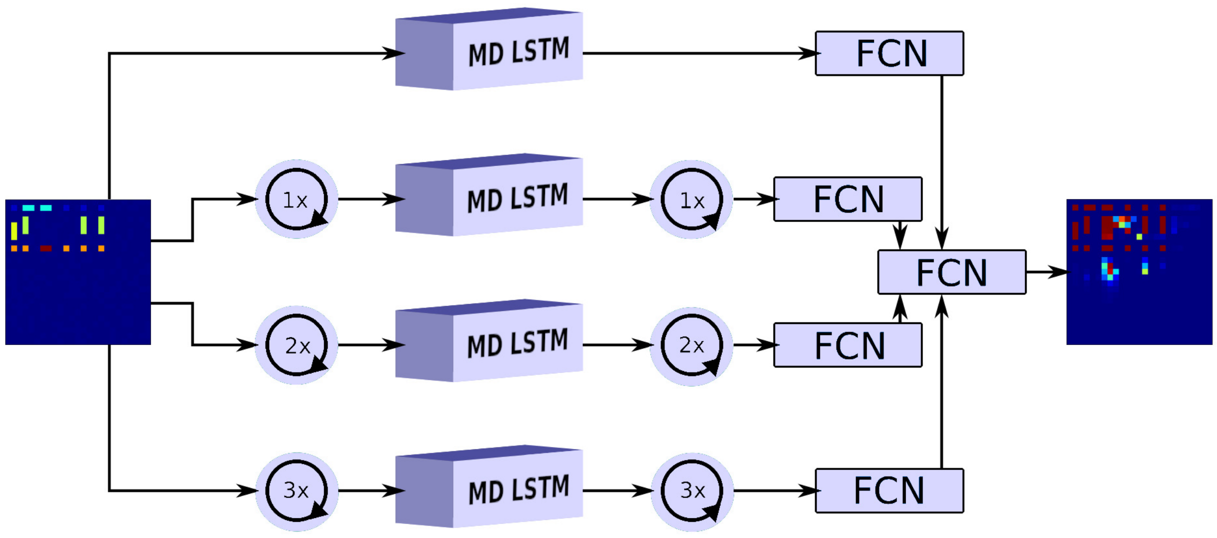


Figure 5: Rotated Multi-Dimensional Long Short-term Memory Network. [Hensel et al., 2021]

4 RECURRENT NEURAL NETWORKS FOR PATTERN COMPLETION AND INPAINTING MASK GENERATION

We consider the ability of RNNs to utilize predictions from the past (previously processed spatial regions) to learn the patterns and symmetries of facade layouts. To this end, we introduce the RMD LSTM for facade completion and object recommendation, and present a comparison of outcomes from QRNN, RMD LSTM (and MD LSTM). In addition, we show how the network output can also be used to create inpainting masks for facade images.

4.1 Multi-Dimensional Long Short-term Memory Network

MD LSTM [Graves et al., 2007] is an RNN architecture that allows for multi-dimensional input by utilizing separate memory connections for each dimension. In a two-dimensional setting, the MD LSTM network computes a value for a spatial cell (x, y) based on values of preceding cells $(x-1, y)$ and $(x, y-1)$. Thus, a directed spatial context is established in the network. The proposed RMD LSTM is composed of four MD LSTM networks.

4.2 Quasi Recurrent Neural Network

The QRNN is not an RNN but a CNN that emulates memory connections with an embedded pooling layer. For time-dependent inputs x_1 to x_T, the convolutional layers compute an output for time step t that consist of three vectors: a candidate vector z_t , a forget vector f_t and an output vector o_t:

$$\begin{aligned} z_t &= \tanh(conv_{W_z}(x_t,\dots,x_{t-k+1})) \\ f_t &= \sigma(conv_{W_f}(x_t,\dots,x_{t-k+1})) \\ o_t &= \sigma(conv_{W_o}(x_t,\dots,x_{t-k+1}))\ . \end{aligned}$$

In the convolution *conv*, a filter kernel size of k is used and the weight vectors are represented by W_z, W_f and W_o. The immediate outputs z_t, f_t, and o_t are then used to compute the hidden states c_t in the pooling layers. Let $c_0 := h_0 := 0$. Then the network output h_t for time step t is computed via

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + (1-f_t)\odot z_t \\ h_t &= o_t \odot c_t\ , \end{aligned}$$

where h_t is the network output for time step t.

The operator $\odot$ denotes element-wise multiplication of the vectors. With the exception of the pooling layer, the QRNN can be easily parallelized, while regular recurrent networks compute intermediate outputs sequentially. Due to efficient convolution and pooling operations, the QRNN is fast and memory efficient.

4.3 Rotated Multi-Dimensional Long Short-term Memory Network

RNNs were developed to solve one-dimensional, time-dependent problems in which information of previous network outputs have to be taken into account for subsequent computations. When dealing with two- or three-dimensional spatial problems, often data of a spatial surrounding has to be considered. But when MD LSTM for two-dimensional input deals with cell (x,y), information of "future" cells $(x+1,y)$ and $(x,y+1)$ have not been computed yet. Especially when MD LSTM is applied to facade completion, missing facade objects in the upper left image region are impossible to add. To overcome this problem, we combine four MD LSTM networks. Each network operates on the input matrix rotated by $k\times 90^\circ$, $k\in\{0,1,2,3\}$, respectively. Thus, each single MD LSTM starts its iterative processing at a different corner of the facade layout. The outputs of the four networks are rotated back to the original orientation. Then they are combined by fully connected layers with sigmoid activation, see Figure 5. We also experimented with maximum pooling layers instead of fully connected layers to combine results. This led to final results which were not significantly different.

Based on the previously described neural networks, we set up a reconstruction workflow. In principle, the networks are applied on matrices obtained from IRL simplification, see Section 3.3. Their output is a matrix that contains probability values for cells being labeled as windows or doors, cf. Figure 1. If a probability exceeds a threshold value of 0.5, the cell is interpreted as a recommended object. The object is equipped with a bounding box by using the cell coordinates of the IRL. However, this could lead to multiple boxes that cover a single object. Such boxes are merged in a post-processing step.

It turned out that the results could be significantly improved by helping the networks with additional layout information, see Section 6. To this end, we did not use the binary matrices as inputs but replaced label values 1 that indicate facade objects with cluster numbers representing similar facade objects belonging to the same floor. To this end, we first group neighboring entries of 1 labels. If some groups are positioned in exactly the same rows and possess the same number of columns, then they belong to the same cluster. We enumerate all clusters and use the numbers as new matrix entries. Clustering is only applied to input data, ground truth and network output still consist of probability values.

With minor adjustments, this workflow is also able to automatically generate inpainting masks. By subtracting the binary version of the input matrix, we filter out the added windows and doors. The regions covered by these objects are likely to be occluded in the given facade image. Thus, a mask is generated by drawing filled circles at the positions of these objects. Positions and diameters of circles are obtained from the IRL coordinates. The radius r of each circle is calculated by

$$r = 1.5\cdot\frac{d}{2},$$

where d is the diagonal of the IRL cell. The factor 1.5 is necessary because occlusions typically do not end at cell boundaries.

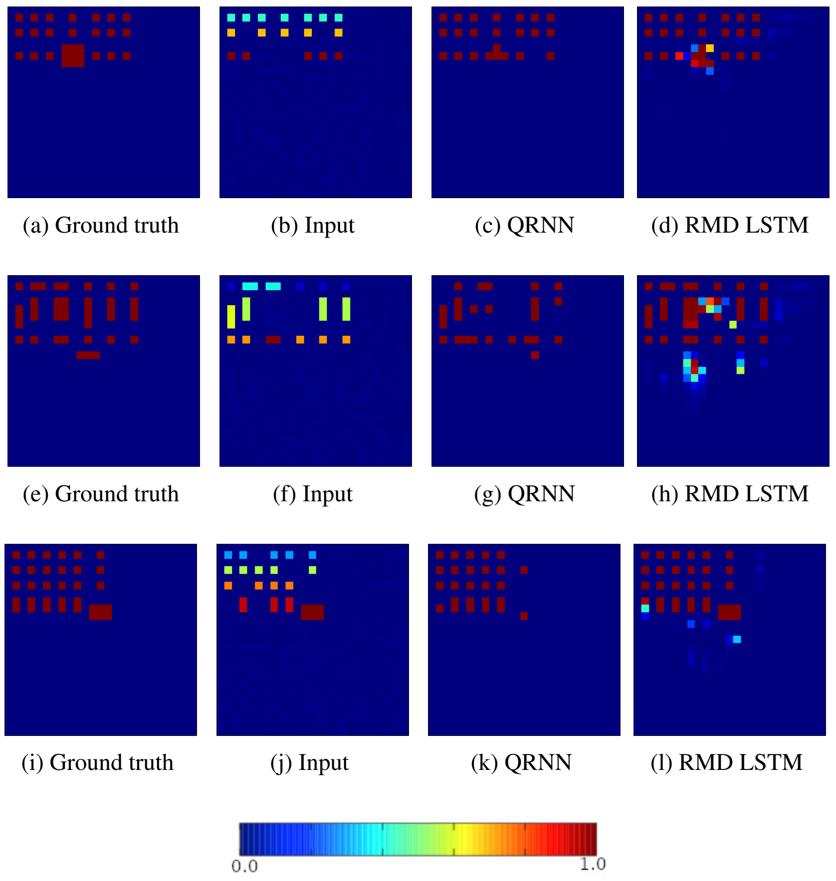


Figure 6: Network output consisting of 25 × 25 confidence values (0 = blue, 1 = red) for three facades of the Graz50 dataset: images *(a)* to *(c)* represent the ground truth, images *(d)* to *(f)* show the input, images *(g)* to *(i)* present the QRNN output, and images *(j)* to *(l)* show the RMD LSTM results. [Hensel et al., 2021]

5 NETWORK TRAINING AND TESTING

To implement, train and test the networks, we used Tensorflow [Abadi et al., 2016]. Our source code is freely available[2].

We trained the networks with a batch size of 16 and over 20,000 batches. Training was performed with randomly removed window and door rectangles from facade layouts of the CMP dataset whereas the original layouts served as the ground truth. This led to 320,000 distorted input matrices. To avoid overfitting, we added low-amplitude noise to background entries of the input, i.e., to the zeros. Training on an NVIDIA P6000 GPU lasted about a full day for MD LSTM and RMD LSTM whereas the QRNN could be trained within 3 hours.

The networks were trained with an equal number of iterations by using the Adam optimizer and a Mean Squared Error loss function.

We tested the trained networks on matrices derived from the Graz50 dataset [Riemenschneider et al., 2012], see Section 3.1.

6 RESULTS

In this Section, we describe results qualitatively and quantitatively. In addition, we show inpainting examples based on masks that were generated with the RMD LSTM as described in Section 4. We also describe the limitations of our approach with some examples.

[2]https://github.com/SimonHensel/LSTM-Facade-Completion

Table 1: Testing Scores.

(a) Comparison of network results on binary input matrices (without cluster labels).

	Acc.	Prec.	Rec.	**IoU**
START	0.938	1.000	0.682	0.682
MD LSTM	0.980	0.913	0.787	0.732
QRNN	0.973	0.888	0.720	0.664
RMD LSTM	0.979	0.907	0.785	0.726

(b) Comparison of QRNN and RMD LSTM on input data labeled by clustering.

	Acc.	Prec.	Rec.	**IoU**
START	0.938	1.000	0.682	0.682
QRNN	0.969	0.901	0.641	0.604
RMD LSTM	**0.984**	**0.925**	**0.832**	**0.779**

6.1 Layout Completion

All experiments were performed on the Graz50 dataset. To be able to compare results, we computed a fixed test dataset with 10,000 facade layouts by removing 20% of facade objects, see Section 3.1.

We begin with a description of quantitative results and discuss qualitative results later. Tables 1a and 1b compare results with and without pre-clustering, cf. Section 4. The output of the neural networks consists of a probability map of confidence values. Object probabilities below a threshold of 0.5 were considered as background. For calculating evaluation scores, we counted every matrix entry in one of four sums: true positive (tp), false positive (fp), true negative (tn) or false negative (fn). Tables 1a and 1b show the average values of

$$\begin{aligned}
\text{Accuracy (Acc.)} &= \frac{\mathrm{tp}+\mathrm{tn}}{\mathrm{tp}+\mathrm{tn}+\mathrm{fp}+\mathrm{fn}}\\
\text{Precision (Prec.)} &= \frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fp}}\\
\text{Recall (Rec.)} &= \frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fn}}\\
\text{IoU} &= \frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fp}+\mathrm{fn}}\ .
\end{aligned}$$

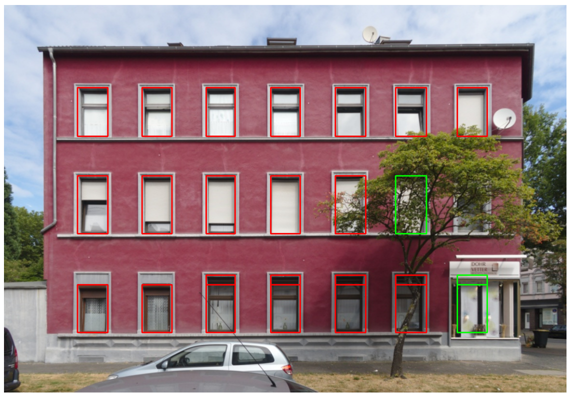
The Intersection over Union (IoU) does not consider the correct classifications of the large background class and thus is especially meaningful. All scores were computed separately for each input and output pair. Afterwards, the arithmetic mean was computed.

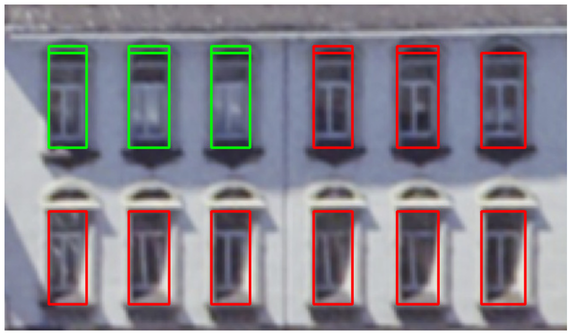
To determine whether the networks were able to improve the facade layouts, we also calculated the scores directly on the network input without applying the networks. The outcomes are listed in the rows in Table 1 denoted by "START". The precision is equal to 1 because no false positives are present in the randomly generated input data.

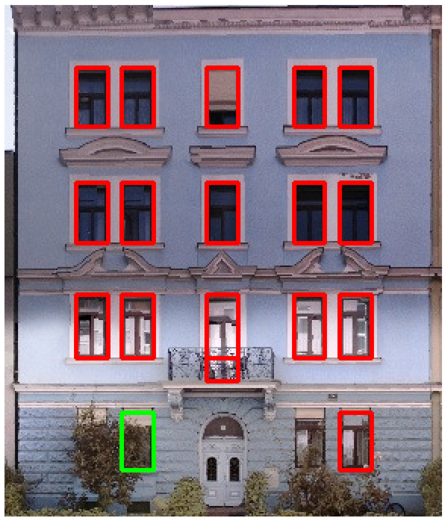
As it can be seen in Table 1a, the RMD LSTM produces better results than the QRNN in terms of accuracy, recall and IoU. Figure 6 illustrates how probability values of recommendations look like. Although we trained the networks on the segregated CMP dataset, RMD LSTM was able to also complete Graz50 layouts. QRNN was not able to achieve similar results.

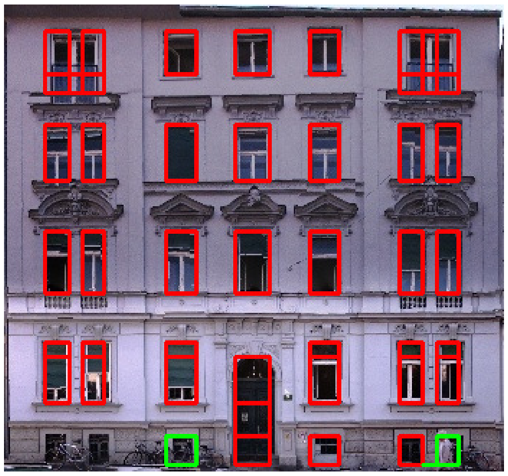
Figure 7 shows examples of RMD LSTM network results on the Scaled YOLOv4 data. Whereas the examples (a), (b), (h), and (i) in Figure 10 were completed within a small margin of error, there were some difficulties with objects that are composed of more than one IRL cell. The network also adds door-shaped objects at wrong positions.

Besides testing on Graz50 data, we also tested on training data. Here, QRNN outperformed the RMD LSTM network. Compared with test data, the IoU of QRNN increased by 0.18. If this is not an effect of overfitting and if the style of facades is known and if the size of training data for this given style is sufficient, then QRNN might be superior to RMD LSTM.


(a)


(b)


(c)


(d)

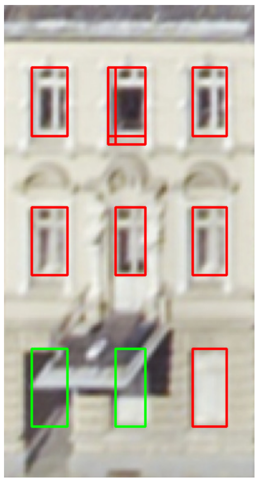

(e)

Figure 7: Detection results of the Scaled YOLOv4 network are shown in red. Objects added by the RMD LSTM are annotated with green boxes.

Furthermore, experiments were also conducted with the GridLSTM network [Kalchbrenner et al., 2015]. This network showed better results for language and text translation tasks than certain LSTM networks due to a grid of multi-way interactions. However, this grid increases memory requirements such that we were not able to apply the network with the same hyperparameters as the other networks. We had to reduce both the batch size and the number of hidden units. But under these restrictions, the network wrongly classified everything as background.

6.2 Inpainting

Besides improving facade object detection, the presented network architecture can also be used to automatically generate inpainting masks, see Section 4. We used these masks to replace occluded or distorted image regions with fitting textures.

We detected windows and doors with the Scaled YOLOv4 network. Then detection results were completed with the RMD LSTM network such that inpainting masks could be generated based on the added objects. The regions defined by the masks were filled with the DeepFillv2 network [Yu et al., 2019]. Figure 8 shows two examples. There is no occlusion in the second example image so that no correction is required. Additional information is required to exclude such images from automatic procession.

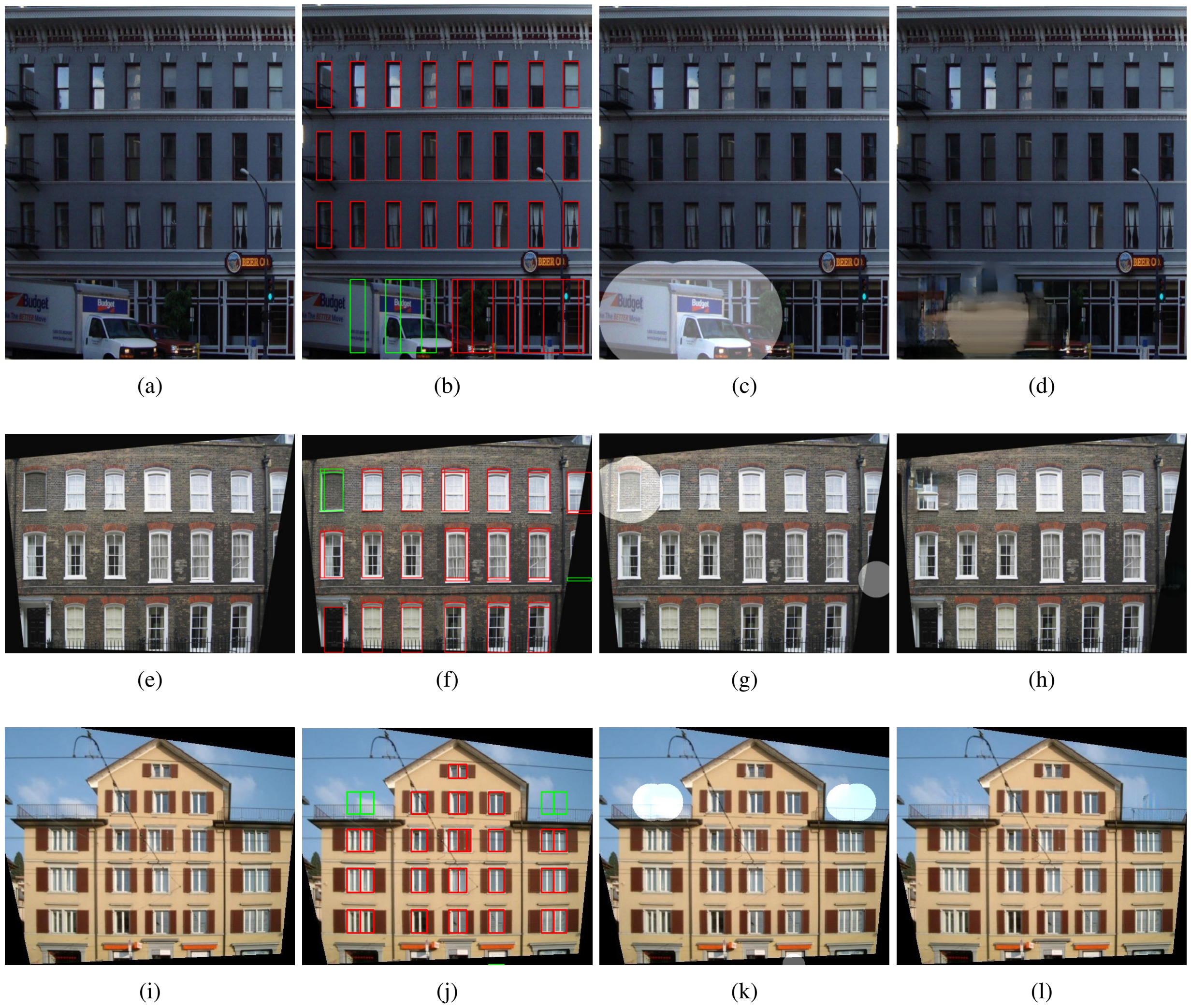


Figure 8: Examples of generated inpainting masks and inpainting results.

6.3 Limitations

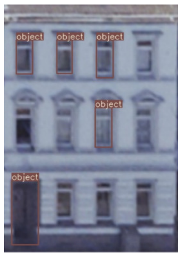
Large regions without detected windows and doors make it difficult to find an IRL that represents the entire facade structure. Two examples of this problem are shown in Figure 9 (a)-(d) and (e)-(h). Another consequence of the sole reliance on object detection is that the LSTM networks do not take pixel information into account. For example, the missing window in Figure 9 (i)-(l) has a different size and shape than the other windows.

Results shown in Figures 9 and 10 indicate that improvements are still possible. The main cause for detection errors are objects that are represented by more than one matrix entry. This occurs in some rare cases and is therefore troublesome for learning. Most often, doors are incomplete or mistaken for a window, see examples in Figure 10 (a), (d) and (j).

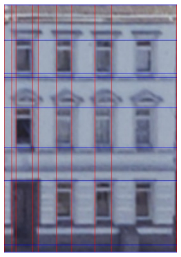
A major limitation in the application of LSTMs is their high memory usage. MD LSTM and RMD LSTM allocate up to 23 GB of GPU memory, depending on the number of hidden units used.

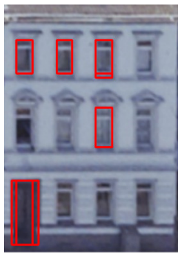
7 CONCLUSION AND FUTURE WORK

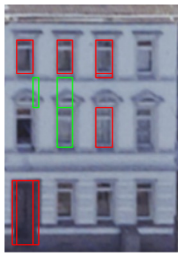
Experiments with the RMD LSTM showed that LSTMs are a suitable means to fill gaps in facade layouts. RMD LSTM outperformed the original MD LSTM and QRNN. It increases the IoU by 14%. An advantage of such deep learning methods over grammar-based algorithms is that no rules have to be defined explicitly.

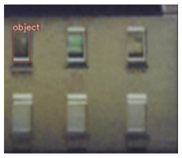


(a) Object detection

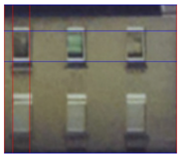

(b) Lattice

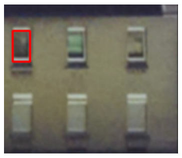

(c) Network Input

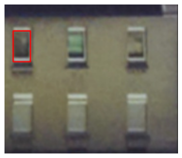

(d) Completion

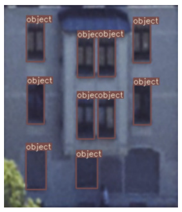


(e) Object detection

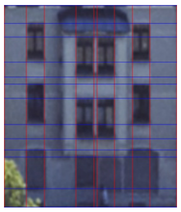

(f) Lattice

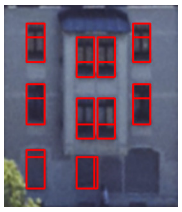

(g) Network Input


(h) Completion



(i) Object detection


(j) Lattice


(k) Network Input

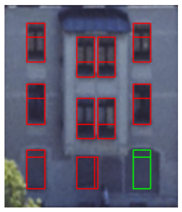

(l) Completion

Figure 9: Examples that showcase the limitations of our workflow presented in Figure 1. Figures (a)-(h) show the effects of an insufficient object detection. Figures (i)-(l) show the effects of separating facade structure knowledge and image information.

So far, the training of neural networks was limited to windows and doors. However, other facade objects could be treated in a similar way. Future improvements should also target the reduction of memory consumption and the other problems listed in Section 6.3.

The current workflow works best if the detected objects are distributed over the entire image. An incomplete IRL resulting from insufficient detection results could be improved by detecting patterns present in the IRL.

References

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). TensorFlow: A system for Large-Scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA. USENIX Association.

[Arjovsky et al., 2017] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR.

[Bradbury et al., 2016] Bradbury, J., Merity, S., Xiong, C., and Socher, R. (2016). Quasi-recurrent neural networks. *arXiv arXiv:1611.01576.*

[Chen et al., 2020] Chen, J., Yi, J. S. K., Kahoush, M., Cho, E. S., and Cho, Y. K. (2020). Point cloud scene completion of obstructed building facades with generative adversarial inpainting. *Sensors*, 20(18):5029.

[Chen et al., 2018] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818.

[Dai et al., 2013] Dai, D., Riemenschneider, H., Schmitt, G., and Van Gool, L. (2013). Example-based facade texture synthesis. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1065–1072.

[Dehbi et al., 2016] Dehbi, Y., Staat, C., Mandtler, L., Pl, L., et al. (2016). Incremental refinement of facade models with attribute grammar from 3D point clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:311.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

[Graves et al., 2007] Graves, A., Fernández, S., and Schmidhuber, J. (2007). Multi-dimensional recurrent neural networks. *CoRR*.

[Gröger et al., 2007] Gröger, G., Kolbe, T. H., and Czerwinski, A. (2007). OpenGIS CityGML Implementation Specification (City Geography Markup Language). *Open Geospatial Consortium Inc, OGC*.

[He et al., 2017] He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[Hensel et al., 2019] Hensel, S., Goebbels, S., and Kada, M. (2019). Facade reconstruction for textured LoD2 CityGML models based on deep learning and mixed integer linear programming. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5:37–44.

[Hensel et al., 2021] Hensel, S., Goebbels, S., and Kada, M. (2021). LSTM architectures for facade structure completion. In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP,*, pages 15–24. INSTICC, SciTePress.

[Hu et al., 2020] Hu, H., Wang, L., Zhang, M., Ding, Y., and Zhu, Q. (2020). Fast and regularized reconstruction of building facades from street-view images using binary integer programming. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2020:365–371.

[Huang et al., 2014] Huang, J.-B., Kang, S. B., Ahuja, N., and Kopf, J. (2014). Image completion using planar structure guidance. *ACM Transactions on graphics (TOG)*, 33(4):1–10.

[Kalchbrenner et al., 2015] Kalchbrenner, N., Danihelka, I., and Graves, A. (2015). Grid long short-term memory. *arXiv:1507.01526*.

[Kottler et al., 2020] Kottler, B., Bulatov, D., and Zhang, X. (2020). Context-aware patch-based method for façade inpainting. In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP*, pages 210–218.

[Lin et al., 2017] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988.

[Mehra et al., 2020] Mehra, S., Dogra, A., Goyal, B., Sharma, A. M., and Chandra, R. (2020). From textural inpainting to deep generative models: An extensive survey of image inpainting techniques. *Journal of Computer Science*, 16(1):35–49.

[Mtibaa et al., 2020] Mtibaa, F., Nguyen, K.-K., Azam, M., Papachristou, A., Venne, J.-S., and Cheriet, M. (2020). LSTM-based indoor air temperature prediction framework for HVAC systems in smart buildings. *Neural Computing and Applications*, pages 1–17.

[Nazeri et al., 2019] Nazeri, K., Ng, E., Joseph, T., Qureshi, F. Z., and Ebrahimi, M. (2019). Edgeconnect: Generative image inpainting with adversarial edge learning. *arXiv:1901.00212*.

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Riemenschneider et al., 2012] Riemenschneider, H., Krispel, U., Thaller, W., Donoser, M., Havemann, S., Fellner, D., and Bischof, H. (2012). Irregular lattices for complex shape grammar facade parsing. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1640–1647.

[Salehinejad et al., 2017] Salehinejad, H., Sankar, S., Barfett, J., Colak, E., and Valaee, S. (2017). Recent advances in recurrent neural networks. *arXiv:1801.01078*.

[Sherstinsky, 2020] Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404:132306.

[Tan et al., 2020] Tan, M., Pang, R., and Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790.

[Teboul et al., 2011] Teboul, O., Kokkinos, I., Simon, L., Koutsourakis, P., and Paragios, N. (2011). Shape grammar parsing via reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2273–2280. IEEE.

[Tobin et al., 2017] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30.

[Tyleček and Šára, 2013] Tyleček, R. and Šára, R. (2013). Spatial pattern templates for recognition of objects with regular structure. In Weickert, J., Hein, M., and Schiele, B., editors, *Pattern Recognition*, pages 364–374, Berlin, Heidelberg. Springer.

[Wang et al., 2021] Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2021). Scaled-YOLOv4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13029–13038.

[Wonka et al., 2003] Wonka, P., Wimmer, M., Sillion, F., and Ribarsky, W. (2003). Instant architecture. *ACM Transactions on Graphics (TOG)*, 22(3):669–677.

[Yu et al., 2018] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., and Huang, T. S. (2018). Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5505–5514.

[Yu et al., 2019] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., and Huang, T. S. (2019). Free-form image inpainting with gated convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[Yu et al., 2020] Yu, T., Guo, Z., Jin, X., Wu, S., Chen, Z., Li, W., Zhang, Z., and Liu, S. (2020). Region normalization for image inpainting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):12733–12740.

[Zhang and Wang, 2016] Zhang, D. and Wang, D. (2016). Relation classification: CNN or RNN? In Lin, C.-Y., Xue, N., Zhao, D., Huang, X., and Feng, Y., editors, *Natural Language Understanding and Intelligent Applications. ICCPOL 2016, NLPCC 2016. Lecture Notes in Computer Science*, volume 10102, pages 665–675, Cham. Springer International Publishing.

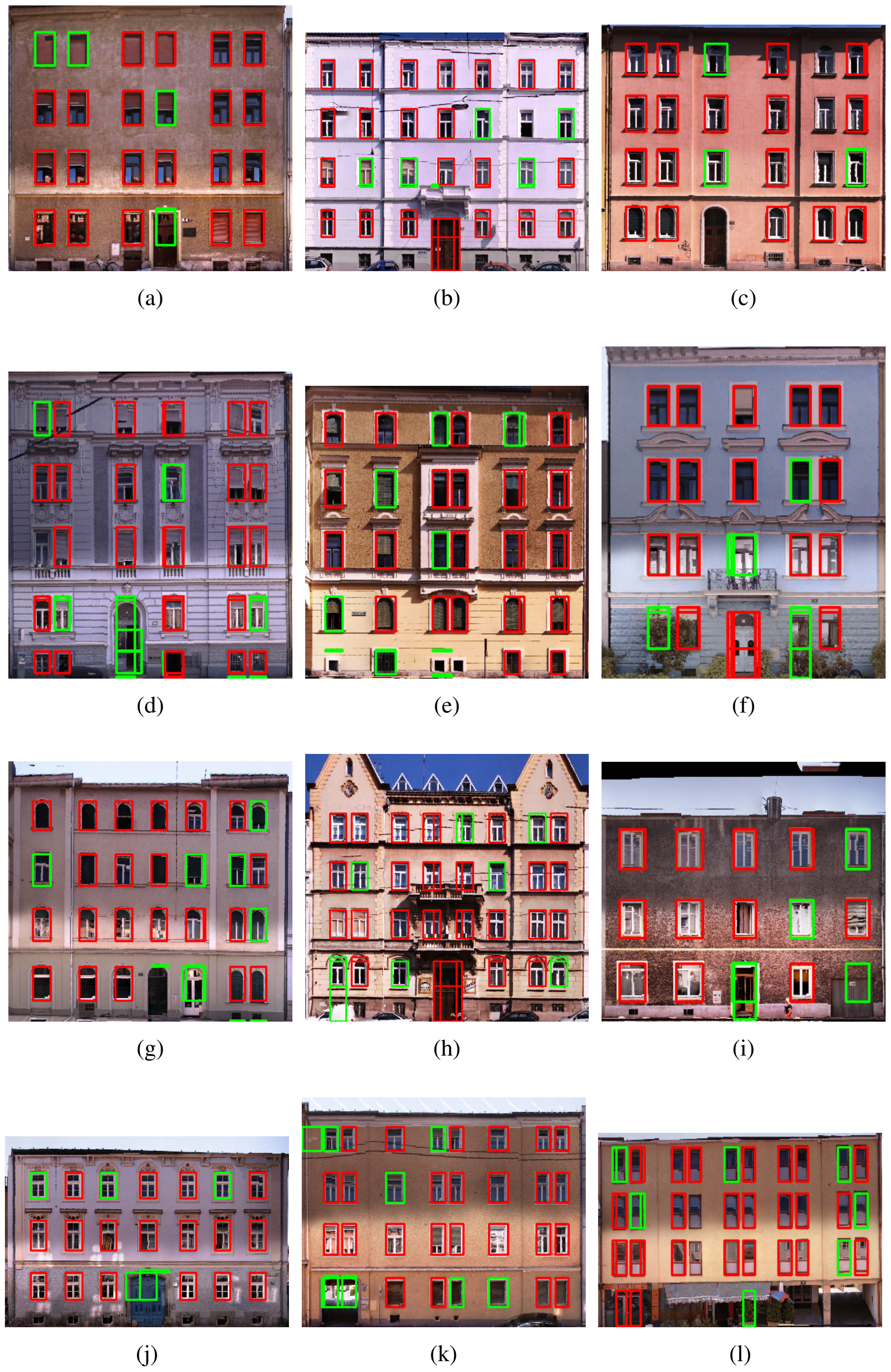


Figure 10: Objects were randomly removed from the ground truth of the Graz50 dataset, see also 3.1. Remaining objects are annotated with red boxes. The RMD LSTM network added the missing objects within a small margin of error. Added objects are marked with green boxes. [Hensel et al., 2021]