ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Segmentation of Geometrical Shapes in Aerial Images

Master's Thesis

Zuoyue Li

April 2018

Supervisors: Prof. Dr. T. Hofmann, Dr. A. Lucchi, Dr. J. D. Wegner
Department of Computer Science, ETH Zürich

Abstract

This example thesis briefly shows the main features of our thesis style, and how to use it for your purposes.

Acknowledgment

Thanks Mum!

Contents

Chapter 1

Introduction

This chapter mainly provides a brief introduction to the entire project. Section 1.1 gives the background of the project and some basic concepts. Section 1.2 defines the problems of this project to be solved. Section 1.3 gives a brief introduction to the proposed new model. Section 1.4 gives the structure of this thesis for the convenience of readers.

1.1 Background

In this section, the background of this project is introduced. Subsection 1.1.1, 1.1.2 and 1.1.3 focus on aerial images, image segmentation and the segmentation with geometry respectively.

1.1.1 Aerial Image

An aerial image is a projected image which is "floating in air", and cannot be viewed normally. It can only be seen from one position in space, often focused by another lens (from Wikipedia). In this project, aerial images generally refer to satellite images, which are taken from satellites flying around the earth. These kind of images have an extremely wide range of applications in the field of geographical surveying and mapping. It can not only clearly depict the terrain but can also show the structure and the layout of the city. Furthermore, it can also provide services such as land use status and remote sensing monitoring.

Nowadays, many buildings are constantly being updated in real life and many landscapes are changing with natural activities. Therefore, electronic maps need to be updated accordingly, and aerial images can provide those maps with important references. Figure 1.1 shows two examples of aerial images.

(a) an area of Zurich (b) an area of Chicago

Figure 1.1: Example of two satellite images.

1.1.2 Image Segmentation

In computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super-pixels). The goal of segmentation is to simplify and change the representation of an image into something that is more meaningful and easier to analyze (from Wikipedia).

In fact, image segmentation is a process of labelling each pixel in an image. Pixels with the same label are generally similar in the metric of certain visual characteristics, such as color, brightness or texture. Adjacent areas are usually very different under this metric. Specifically, for segmentation in aerial images, usually what we do is to label each pixel as buildings, roads or background. Figure 1.2 shows an example.

label012

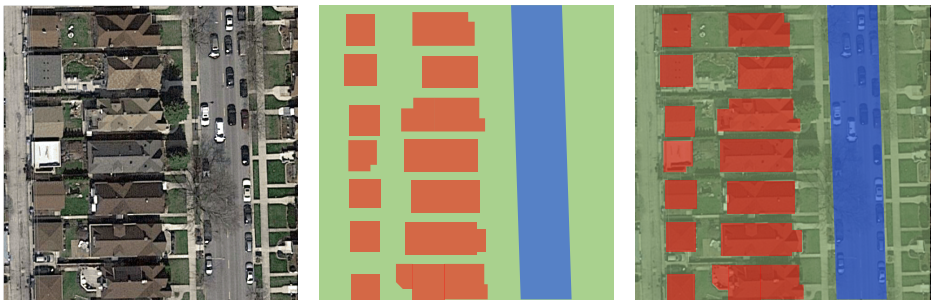
1.1.3 Geometrical Shape

- - - - [10,11,12,13,14]CNNpixel wise

1.2 Problem Definition

shape

Instance segmentation is challenging because it requires the correct detection of all objects in an image while also precisely segmenting each instance. It therefore combines elements from the classical computer vision tasks of object detection, where the goal is to classify individual ob- jects and localize



(a) original image (b) segmentation result (c) alpha composition

Figure 1.2: Example of segmentation in aerial image. The original image (a) is an area of Chicago. In (b), red color denotes buildings, blue color denotes roads, green color denotes background. (c) is (a) overlaid by (b) for clearness.

each using a bounding box, and semantic segmentation, where the goal is to classify each pixel into a fixed set of categories without differentiating object in- stances.

1.3 Focus of This Work

region-based PolygonRNNMask R-CNNFPNPolygonRNNFPNRoIPolygonRNNfind geometric shape

1.4 Thesis Organization

2 bboxprevious thesismask r-cnnpolygon 3Faster R-CNN4567

Following common terminology, we use object detection to denote detection via bounding boxes, not masks, and semantic segmentation to denote per-pixel classification without differentiating instances. Yet we note that instance segmentation is both semantic and a form of detection.

Chapter 2

Related Work

In this chapter, some related work are illustrated in detail. Section 2.1 presents two previous theses and points out their deficiencies with regard to our problem. Section 2.2 introduces two recent models, PolygonRNN and Mask R-CNN. Section 2.3 gives the summary of these models and further discusses the feasibility of applying these models to our problem.

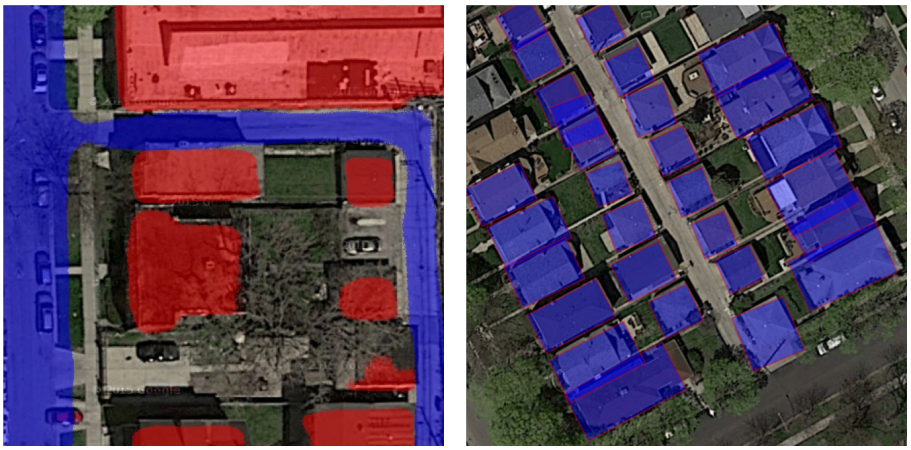
2.1 Previous Theses

There are two previous theses related to our problem, which are shown as follows.

Image Segmentation in Pixel Level In this thesis, our problem is treated as a pixel-wise image segmentation problem. The student employs a kind of deep neural network, U-Net, and make modification to classify each pixel in the aerial image as different labels, i.e. buildings, roads or background. The output of the model is a labelled mask. An example is shown in figure 2.1(a). We can see that the model can well distinguish between buildings and roads. However, there are also some limitations in this method: (1) we do not know which pixel belongs to which building, meaning that there is no instance segmentation here; (2) the buildings cannot be detected unless we do further pixel connectivity detection; (3) there are no geometrical shapes of the objects. Therefore, this model cannot be fully used to solve our problems.

Rotated Bounding Box Another thesis utilizes the RPN (Region Proposal Network) part in Faster R-CNN and adds a branch for orientation degree. It predicts the rotated bounding boxes to 'capture' each building instance in an aerial image. Thus, our problem here becomes a parameters regression problem, which aims at finding out the center coordinates, width, height, and the rotation degrees of the bounding box. The model used here is the

modified RPN. The role of RPN part in original Faster R-CNN is to find RoIs (Regions of Interest), which are generally represented as bounding boxes or rectangles. This kind of representation can be very suitable for horizontal or vertical buildings. But we know that buildings are not always regular, many of them are inclined in the image. Therefore, in this thesis project, the RPN is modified and another branch for rotating the bounding boxes is added. This can make the boundary of the bounding box closer to the building. Figure 2.1(b) shows an example output. We can see from the figure that each building instance can correspond to a rotated bounding box, and can be also limited tightly to the box. This methods can detect buildings very well, but the geometrical shapes found are limited to rectangles and cannot be more precise.



(a) pixel-wise labelled mask (b) rotated bounding boxes

Figure 2.1: Example outputs of two previous theses. In (a), the pixels with red color refer to buildings, while the pixels with blue color refer to roads. In (b), each bounding box relates to one instance.

In short, the analysis and discussion of the two theses above show that although they have their own advantages and the effects are still good, they cannot fully reflect the geometrical shapes.

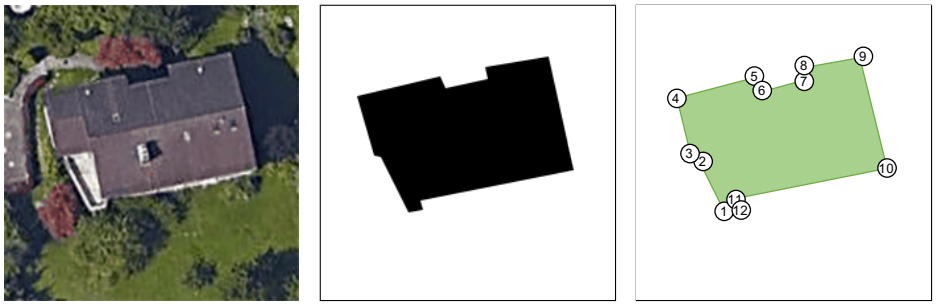
2.2 Recent Models

In this section, two recent models, PolygonRNN and Mask R-CNN are illustrated (see subsection 2.2.1 and 2.2.2 respectively). PolygonRNN focuses on the segmentation of geometrical shape and the FPN (Feature Pyramid Network) in Mask R-CNN can be used for the detection of buildings.

2.2.1 PolygonRNN

PolygonRNN aims at finding geometrical shape for an object instance, given the bounding box. The model is originally proposed for speeding up labeling the ground truth since it can achieve semi-automatic annotation of object instances, but the model can be used for single instance segmentation as well.

Most current methods regard the instance segmentation problem as a pixel-wise classification problem, generally labeling each pixel as object or background (see figure 2.2(b) for example). Different from this traditional way, the paper treats the segmentation task as a polygon prediction problem. In particular, PolygonRNN takes the image of instance as input and sequentially produces vertices of the polygon outlining the object (see figure 2.2(c) for example).



(a) an example building (b) pixel-wise mask (c) polygon prediction

Figure 2.2: Comparison of pixel-wise mask and polygon. (a) is the original image containing a building. (b) is the target mask of traditional pixel-wise instance segmentation, (c) is the desired prediction of polygon, of which the vertices are numbered.

We know that predicting a polygon is equivalent to predicting each of its vertices. Thus, the paper regards polygon as a series of vertices, and uses RNN as the model to make coherent prediction. RNN is very powerful when data is related to time series as it can carry complex information about the history. In our case, the prediction of each vertex is dependent on the position of its two previous vertices. The paper also mentioned that another advantage over traditional methods is that RNN can capture the shape of the object even in ambiguous cases like shadows and saturation.

In short, we hope that PolygonRNN can be used to solve our problems, extracting geometric shapes of buildings in aerial images, since the buildings in such images can be also regarded as polygons. For more details of the architecture, please refer to section 3.1.

2.2.2 Mask R-CNN

Mask R-CNN is a general framework for object instance segmentation. It is the latest model in the R-CNN family, which includes R-CNN, Fast R-CNN, Faster R-CNN as well. Mask R-CNN can efficiently detect objects in an image and generate a high-quality segmentation mask for each instance simultaneously. Figure 2.3 shows example results of Mask R-CNN. We can see from the figure that this model achieves excellent results. Figure 2.4 shows the simplified structure of Mask R-CNN. Each part of the structure are illustrated as follows.

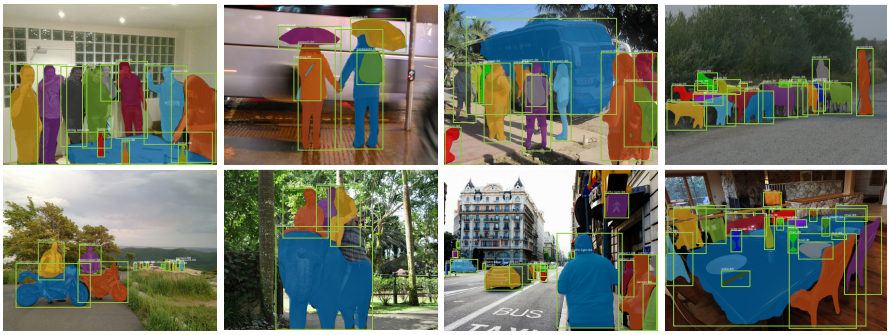


Figure 2.3: Example results of Mask R-CNN.

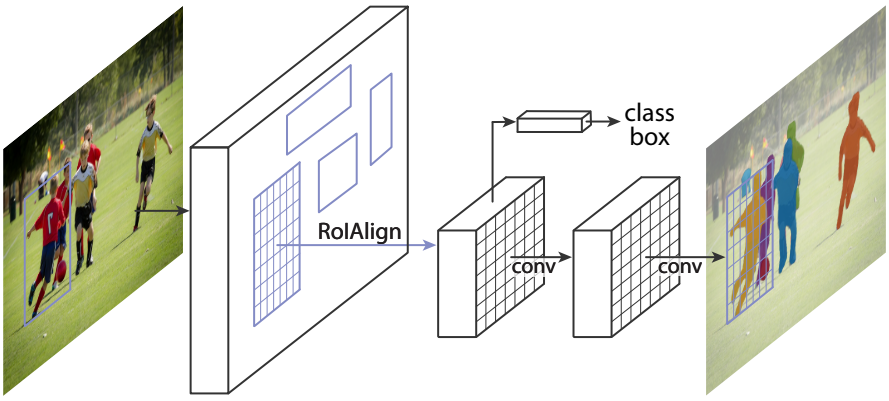


Figure 2.4: Simplified model structure of Mask R-CNN. Feature Pyramid Network locates at the first layer in the figure.

Feature Pyramid Network The original image is first passed through the FPN (Feature Pyramid Network), which locates at the first layer in figure 2.4. Just like the RPN mentioned in section 2.1, FPN is also a kind of network for finding RoIs. Actually, FPN is an upgraded version of RPN used in both Faster R-CNN and the previous thesis. Compared with RPN, FPN solves the

problem of multi-scale detection and introduces feature pyramid, and thus improves the accuracy of detection, especially for small objects in the image.

Classification Branch The classification branch exists from the earliest R-CNN to the present Mask R-CNN. It classifies each object in RoI found by FPN as different classes and gives probability, using fully connected layers. However, this branch is not relevant to our problem, since currently we only interested in buildings rather than kinds of different objects in an aerial image.

Mask Branch Mask R-CNN extends Faster R-CNN by adding the mask branch for predicting an object mask on each RoI in parallel with the existing classification branch. This branch uses small FCN (Fully Convolutional Network) for the pixel-wise semantic segmentation. The mask branch should be relevant to our problem, but the mask it predicts is still in the pixel level instead of polygon.

In short, Mask R-CNN can surpass prior instance segmentation results, but it cannot give any geometrical information of the objects in an image. Therefore, FPN is the only part of Mask R-CNN, which can be utilized to solve our problems. We hope that FPN can make correct detection of all buildings and localizing each using a bounding box. For more details of the architecture of FPN, please refer to section 3.2.

2.3 Motivation

Table 2.1 makes a summary for all models mentioned in sections 2.1 and 2.2. From the table we can conclude that none of these models meets our requirements. Thus, combination of two or even more models becomes necessary.

Table 2.1: Summary of Related Models.

Model	Localization	Geometrical Shape	Classification
Modified U-Net	Limited[1]	No	**Yes**
Modified RPN	**Yes**	Limited[2]	No
PolygonRNN	No	**Yes**	No
Mask R-CNN	**Yes**	No	**Yes**
FPN	**Yes**	No	No
What We Want	**Yes**	**Yes**	N/A [3]

[1] As mentioned in section 2.1, localization requires further pixel connectivity detection.

[2] As mentioned in section 2.1, only rotated bounding boxes are found.

[3] As mentioned in section 1.2, segmentation of roads is currently not considered in our project, thus the cell here shows 'N/A'.

After observation, we propose a possible approach, simply replacing the mask branch in Mask R-CNN with PolygonRNN so that the modified model can predict polygon rather than pixel-wise mask for each RoI. This model takes advantages from both models, thus it can be applied to our problem. In practice, we just combine FPN and PolygonRNN and name it a new model R-PolygonRNN (Region-based PolygonRNN), of which the architecture is shown in detail in section 3.3.

Chapter 3

Model Architecture

In this chapter, the architectures of several models are presented. Section 3.1 gives a comprehensive explanation to the structure of PolygonRNN, while section 3.2 looks into the FPN part of Mask R-CNN. Considering our problem, we combine PolygonRNN and FPN part of Mask R-CNN together and come up with a new model, which is called R-PolygonRNN (Region-based PolygonRNN, see section 3.3). In theory, the proposed model can find out the bounding boxes of buildings within an aerial image and give geometrical shape for each building.

3.1 PolygonRNN

PolygonRNN is the core model for finding geometrical shapes in this project. Figure 3.1 shows the simplified structure of PolygonRNN. The CNN part (see subsection 3.1.1) can capture image features through multilayer convolutions and max pooling, which is then fed into the RNN part (see subsection 3.1.2), which can sequentially predict spatial location of the new vertex with the highest probability at each time step.

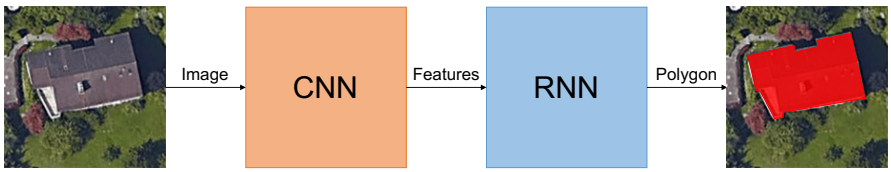


Figure 3.1: Simplified structure of PolygonRNN.

3.1.1 CNN Part

The CNN part of PolygonRNN uses VGG-16. Actually, VGG-16 is a form of VGGNet, which is very structured and focusing on deepening the neural net-

work without a large number of parameters. It generally believes that deeper networks have stronger expressive capabilities than shallow networks, and can accomplish more complex tasks. It also proven in practice that VGGNet has made great progress in performance compared to its previous network architecture (e.g. AlexNet).

The '16' in VGG-16 means that it is a VGGNet with 16 layers containing parameters (13 convolutional layers and 3 fully connected layers). It has around 138 million parameters in total. Figure 3.2 shows its detailed network structure. From the figure we can see that VGG-16 continuously does convolution with 3×3 small kernels and makes 2×2 max pooling. As the network deepens, the width and height of the image are reduced by half after each max pooling, and the number of channels is also doubly increasing after some convolution.

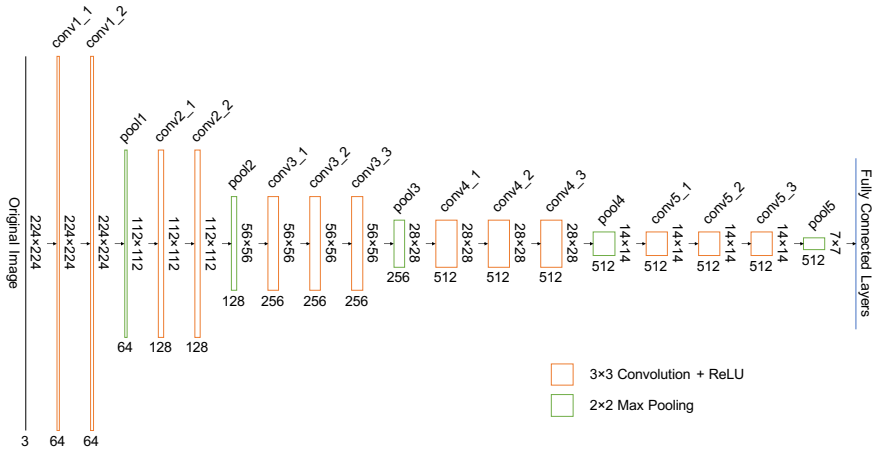


Figure 3.2: VGG-16 architecture. Only convolutional layers and max pooling layers are presented.

VGG-16 was mostly used for image classification before, so after the convolutional layers and max pooling layers there are fully connected layers and softmax layer for the class labels. However, these two kinds of layers are not required for VGG-16 used in PolygonRNN, because the CNN here is working as a feature extractor and mask predictor. Layer `pool5` is omitted as well because of the too low resolution.

Feature Extraction The modified VGG-16 provides RNN with useful features, which are taken from different convolutional and max pooling layers. Specifically, the features are extracted from layers `pool2`, `pool3`, `conv4_3` and `conv5_3`. Note that since the resolution of final features is fixed, when taking features from layer `pool2` and `conv5_3`, it requires another max pooling and upsampling respectively. All of these can be seen in figure 3.3.

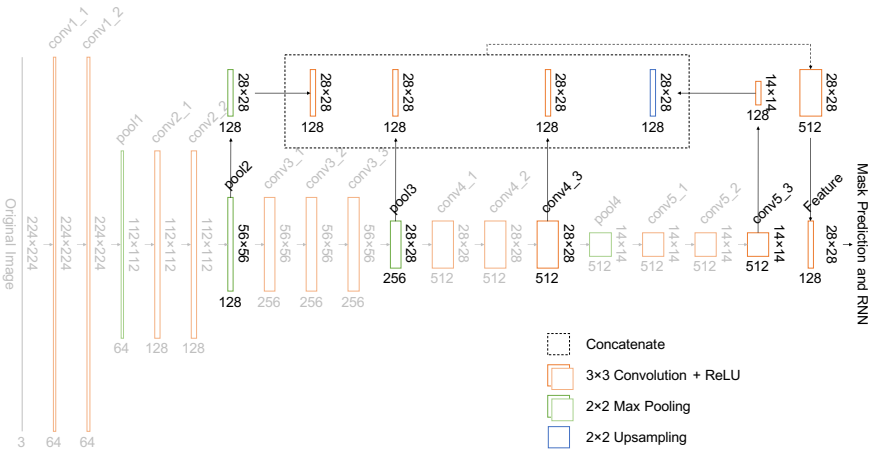


Figure 3.3: Modified VGG-16 architecture in PolygonRNN. In this figure, the highlight layers are used to extract features.

Mask Prediction Another function of the CNN part is to predict masks of boundary and vertices in a low resolution (one eighth of the original). Figure 3.4 shows the mask prediction phase. Different from the ReLU function used in the former convolutional layers, the activation function used here is the sigmoid function. Each entry of the boundary or the vertices mask indicates the probability that the pixel is located in the boundary or is a vertex respectively. The features and two masks are then sent into RNN together.

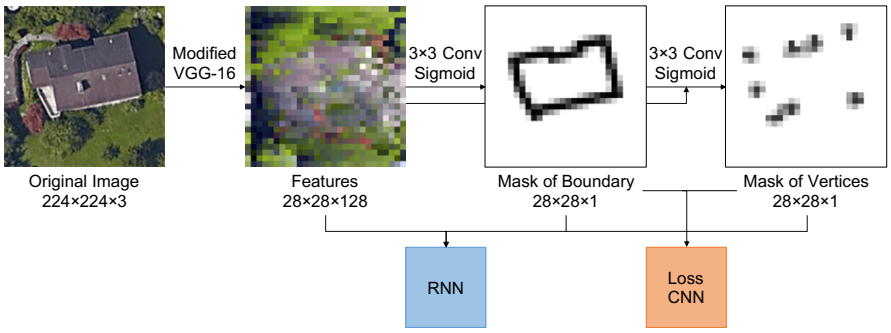


Figure 3.4: Mask prediction of VGG-16. Note that the mask of vertices is obtained by the convolution on the concatenation of the features and the mask of boundary.

3.1.2 RNN Part

The model used for predicting polygon vertices is RNN. We have already mentioned in subsection 2.2.1 that RNN is very powerful when data is related to time series, and in our case, we regards the polygon as a series of vertices.

We know that given two vertices on a polygon in an order (either clockwise or anticlockwise), the third vertex after the two points can be uniquely determined. What RNN here can do is to predict the probability distribution of the next vertex's position in a low resolution when given the history information about the two vertices before, as well as the image features and the position of the starting vertex. This process can be formulated as follows.

$$p(v_t \mid v_{t-1}, v_{t-2}) = F(v_{t-1}, v_{t-2}, v_0, f), \forall t \in \{2, 3, 4, \ldots, T\}, \tag{3.1}$$

where f denotes the extracted features by CNN (including the masks of boundary and vertices), $F(\cdot)$ denotes a function for computing the conditional probability, T denotes the number of vertices of the polygon, v_t denotes the vertex position at t-th prediction. Specifically, the vertex prediction problem can be formulated as a classification problem. The position of v_t can be then quantized to the resolution of output grid, and we can thus use one-hot encoding for v_t's representation.

End Signal Note that the positions for v_t are not only limited to the general output grid, the end signal for the closure detection of the polygon is also embedded in v_t, just like the 'end of sequence' token `<eos>` or `</s>` in the RNN language model. Thus, the number possible assignments of v_t equals to the resolution of the output grid plus one ($28 \times 28 + 1 = 785$ in our case). In order to correctly predict the end signal, the starting vertex v_0 is required for the conditional probability (equation 3.1) calculation, as it tells the model when to finish the prediction phase. If the current prediction is the same as, or very close to the starting vertex v_0, v_t will be forced to raise the end signal, indicating that the entire polygon is close, and the prediction phase is therefore complete. So generally, the end signal works when $t = T$.

Starting Vertex In equation 3.1, two special cases $p(v_0)$ and $p(v_1 \mid v_0)$ are not included yet. These two cases are different from the general case, and should be considered in addition. In particular, we can directly regard the mask of vertices (for example, the rightmost image in figure 3.4) predicted by the CNN as v_0's unnormalized probability distribution $\tilde{p}(v_0)$, and choose the position with the highest probability for v_0's assignment. As v_0 is known, there are typically two options for v_1, one is the next vertex on its left direction and another on right direction. To tackle this problem, we can simply specify the order of the polygon vertices to be fixed, so that v_1 can be uniquely determined. In our project, the order of polygon is set to be anticlockwise.

ConvLSTM The RNN uses ConvLSTM (Convolutional LSTM) cells as the polygon decoder to sequentially predict vertices. For simplicity, we can regard ConvLSTM as the function $F(\cdot)$ in equation 3.1. In fact, the structure of a ConvLSTM cell is almost the same as that of an ordinary LSTM cell, except

that it employs convolution in a 2D image with multiple channels instead of matrices or vectors multiplication. The introduction of ConvLSTM can significantly reduce the the number of parameters when it is compared with a fully connected RNN. The following equations define the computation process within a ConvLSTM cell, which is also visualized in figure 3.5.

$$\begin{bmatrix} f_t \\ i_t \\ g_t \\ o_t \end{bmatrix} = \begin{bmatrix} W_{hf} \\ W_{hi} \\ W_{hg} \\ W_{ho} \end{bmatrix} * h_{t-1} + \begin{bmatrix} W_{xf} \\ W_{xi} \\ W_{xg} \\ W_{xo} \end{bmatrix} * x_t + \begin{bmatrix} b_f \\ b_i \\ b_g \\ b_o \end{bmatrix} = W_h * h_{t-1} + W_x * x_t + b, \tag{3.2}$$

$$c_t = \sigma(f_t) \circ c_{t-1} + \sigma(i_t) \circ \tanh(g_t), \tag{3.3}$$

$$h_t = \sigma(o_t) \circ \tanh(c_t), \tag{3.4}$$

where x_t, h_t, c_t denote the input, the hidden state (or cell output), and the cell state of the cell at time step t respectively, i_t, o_t, f_t denote the states of input, output, and forget gate at time step t respectively, g_t denotes an intermediate variable, $\sigma(\cdot)$, $*$, $\circ$ denote the sigmoid function, convolution, and Hadamard (element-wise) product respectively, W_x and W_h denotes two convolution kernels for x_t and h_t respectively, W_{xf}, W_{xi}, W_{xg}, W_{xo} denote the four components of W_x in the dimension of channels and W_{hf}, W_{hi}, W_{hg}, W_{ho} denote the four components of W_h in the dimension of channels.

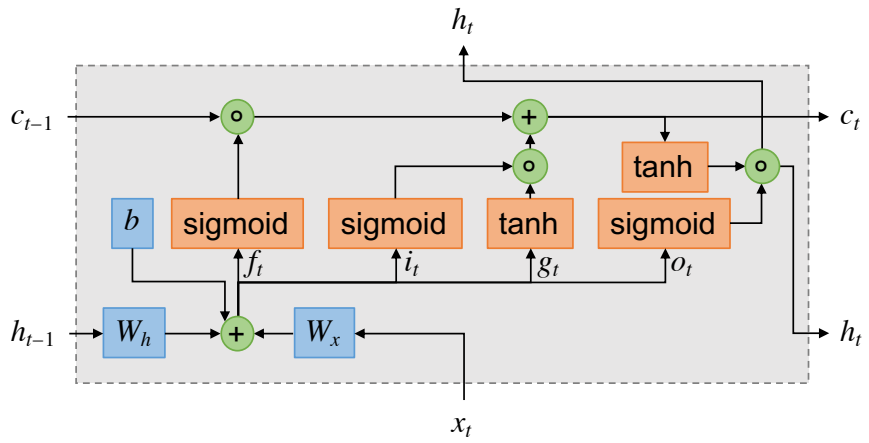


Figure 3.5: Visualization for LSTM cell.

Multilayer RNN PolygonRNN uses multilayer RNN with ConvLSTM cells. The connection between two neighboring layers can be described as follows, which means that the cell input of this layer comes from the cell output of the previous layer.

$$x_t^{(k)} = h_t^{(k-1)}, \forall k \in \{2, 3, \ldots, n\}, \tag{3.5}$$

where n is the number of RNN layers or ConvLSTM cells, k in the superscript denotes the k-th layer. Recall that the initial cell input (i.e. the cell input of the first layer) at time step t consists of the spatial information f from CNN (including the masks of boundary and vertices), the two previous vertices v_{t-1} and v_{t-2}, and starting vertex v_0, here it is formulated as follows.

$$x_t = x_t^{(1)} = v_{t-1} \oplus v_{t-2} \oplus v_0 \oplus f, \tag{3.6}$$

where $\oplus$ denotes the concatenation operation in the dimension of image channel. Thus, the equation 3.1 can be written as follows.

$$y_t = p(v_t \mid v_{t-1}, v_{t-2}) = \text{softmax}(W_y \text{vec}(h_t^{(n)})), \tag{3.7}$$

where W_y denotes the weights of the final fully connected layer, $\text{vec}(\cdot)$ denotes the vectorization function for a matrix. Figure 3.6 shows the entire structure of the RNN part and highlights the process at time step t, using the same notation as the equations 3.5, 3.6 and 3.7.

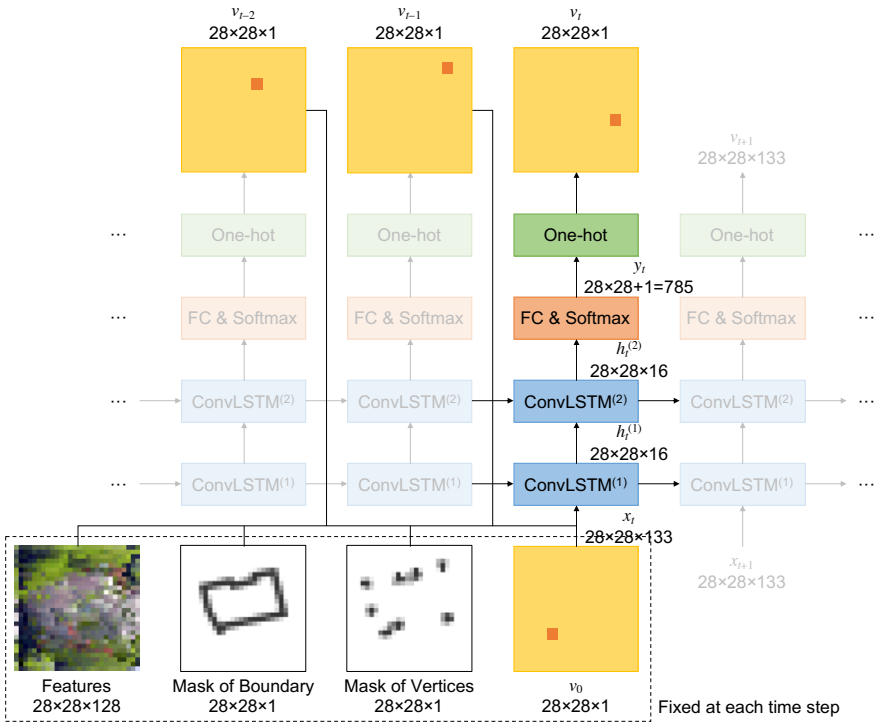


Figure 3.6: Visualization for the time step of the RNN decoder. In this figure, the outputs of the end signal are omitted, and the configuration of the ConvLSTM cell is the same as what is used in the original PolygonRNN paper. Specifically, the paper sets the number of RNN layers to 2 and uses ConvLSTM cells with 3×3 kernel size and 16 channels.

Note that at each time step, the features, the masks of boundary and vertices as well as the one-hot encoding of starting vertex are fixed. Only v_{t-1} and v_{t-2} change over time.

3.1.3 Loss Function

The loss of PolygonRNN consists of two parts, loss of CNN and loss of RNN. The loss function of CNN part uses log loss, since the mask prediction is equivalent to binary classification for each pixel. We also notice that the non-boundary pixels or the non-vertex pixels occupy the majority, the loss function should be further weighted. As for the loss of RNN, the loss function used is cross-entropy loss, because the polygon prediction is equivalent to multiclass classification at every time step.

3.2 Feature Pyramid Network

FPN is the core model for object detection in this project. Object detection problem is exactly multiple bounding boxes regression problem, which finds out multiple RoIs within an image. Subsection 3.2.1 first illustrates problem of single bounding box regression. Subsection 3.2.2 introduces some basic concepts related to anchor, and subsection 3.2.3 further illustrates multiple bounding boxes regression. Finally, subsection 3.2.4 looks into the backbone of FPN, presents how feature pyramid is generated and how FPN can tackle the multi-scale detection problem.

3.2.1 Single Bounding Box Regression

Single bounding box regression can be used for such a scenario, where the target image has only one RoI. Generally, a bounding box can be uniquely determined by four parameters, either box center and box size, denoting (x, y) and (w, h) or the coordinates of the upper left and lower right corners of the box, denoting (l, u) and (r, d). They have following relationships.

$$\begin{bmatrix} x \\ y \\ w \\ h \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -2 & 0 & 2 & 0 \\ 0 & -2 & 0 & 2 \end{bmatrix} \begin{bmatrix} l \\ u \\ r \\ d \end{bmatrix}, \tag{3.8}$$

$$\begin{bmatrix} l \\ u \\ r \\ d \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 2 & 0 & -1 & 0 \\ 0 & 2 & 0 & -1 \\ 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \\ h \end{bmatrix}. \tag{3.9}$$

Therefore, we can regard the problem of finding out single bounding box of RoI as a parameter regression problem. With the image as input, any set of

the four parameters, (x, y, w, h) or (l, u, r, d), can be selected as the target of the regression. Figure 3.7 shows the architecture of the network for the single bounding box regression, using box center and size as outputs.

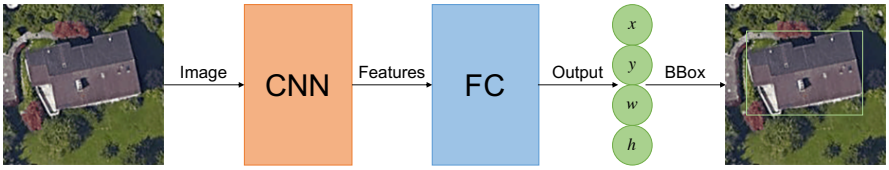


Figure 3.7: Simplified structure of the network for single bounding box regression. In this figure, the output of FC layer refers to the center and size of the bounding box.

In practice, usually the normalized parameters of box center and size rather than the coordinates of two corners are used as training targets. As for the normalization functions, see subsection 3.2.3 for more details.

3.2.2 Anchor and its Properties

Anchor is a basic concept in multiple bounding boxes regression. The following paragraphs introduce the idea of anchor and its properties in detail.

Anchor An anchor refers to an artificially specified bounding box in the original image, regardless of the number of RoIs or where these RoIs locate. An arbitrary rectangular area in the image can be an anchor. For an image with width w and height h, the total number of possible anchors n can be calculated as

$$n = \frac{1}{4} w(w+1)h(h+1). \tag{3.10}$$

Thus, even for a very small image patch with size 224×224, n is already very large, reaching 635.04 million. Figure 3.8 illustrates some examples of anchors in an image with multiple RoIs.

IoU Score Usually IoU (Intersection over Union) score is used to evaluate the similarity between an anchor and a ground truth bounding box. The following equation shows its computation.

$$s = \frac{|A \cap T|}{|A \cup T|} \in [0, 1], \tag{3.11}$$

where s denotes the IoU score, $|\cdot|$ denotes the size of the set, A and T refer to the sets of pixels covered by the anchor and the ground truth bounding box respectively. $s = 1$ means that the anchor and the ground truth bounding box overlap perfectly. Figure 3.9 shows the coverage level under different IoU scores.

(a) Positive (b) Natural (c) Natural (d) Negative

Figure 3.8: Example anchors in image with multiple RoIs. (a)–(d) show four possible anchors in an image with with multiple RoIs, with each showing one anchor. The red and light green rectangles refer to anchors and ground truth bounding boxes respectively. The polarities of these anchors are also shown, using the typical threshold.

(a) IoU = 0 (b) IoU = 0.3 (c) IoU = 0.7 (d) IoU = 1

Figure 3.9: Coverage level of anchor and ground truth bounding box under different IoU scores.

Anchor Polarity Generally, there are several buildings in an aerial image, thus multiple RoIs exist. Based on the IoU scores of an anchor with these ground truth bounding boxes, we can make a judgment on the polarity of this anchor, which is defined as follows.

$$d_i = \underset{j}{\operatorname{argmax}} \operatorname{IoU}(a_i, g_j), \tag{3.12}$$

$$s_i = \operatorname{IoU}(a_i, g_{d_i}), \tag{3.13}$$

$$p_i = \begin{cases} 1, & s_i > t_h, \\ 0, & t_l \leqslant s_i \leqslant t_h, \\ -1, & s_i < t_l, \end{cases} \tag{3.14}$$

where a_i and g_j denote the i-th anchor and j-th ground truth bounding box, s_i, d_i, and p_i denote the highest IoU score that a_i can get, the index of the 'closest' ground truth box of a_i, and the polarity of a_i, t_l and t_h denote two threshold for the polarity judgment, and typically, $t_h = 0.7$, $t_l = 0.3$. An anchor is labeled as positive, natural or negative in case of $p_i = 1$, $p_i = 0$ or $p_i = -1$ respectively. Figure 3.8 also shows anchors with different polarities.

3.2.3 Multiple Bounding Boxes Regression

Multiple bounding boxes regression can be very different from the single one. Since the number of RoIs is unknown, if we still directly use the parameters regression as what we do in case of the single bounding box, then the number of nodes of the FC output layer will not be fixed. Besides, the RoIs in the image usually reflects the local information, usually we cannot directly use all global features obtained by CNN, either.

Anchor Assignment The basic idea for multiple bounding boxes regression is to regress them on positive anchors. Specifically, each RoI is regressed on the anchors which are assigned to it. There are two rules for the assignment: (1) each ground truth bounding box should have at least one anchor (either positive, natural or negative) assigned to it; (2) each positive anchor should be assigned to its 'closest' ground truth bounding box. For those non-positive anchors that have assignments, we also treat them as positive anchors. For details of the matching algorithm, please refer to section A.4.

Classification and Regression on Anchors As a matter of fact, the single bounding box regression can be regarded as regression on the anchor, which is exactly the whole image. Similar to this, we can simply increase the number of nodes in the FC output layer (see figure 3.7) according to the number of anchors. Generally, each anchor requires 6 output nodes, 2 for binary classification and 4 for parameters regression. Specifically, 2 classification nodes output the logits, denoting l_p and l_n, indicating how close the anchor is to an RoI. Thus the predicted probability of an anchor referring to an RoI p^* can be described as

$$p^* = \frac{e^{l_p}}{e^{l_p} + e^{l_n}}, 1 - p^* = \frac{e^{l_n}}{e^{l_p} + e^{l_n}}. \tag{3.15}$$

The rest 4 regression nodes output the refined box information, for recovering the corresponding RoI. Suppose we have a fixed positive anchor, which is assigned to a ground truth bounding box. The refined (or normalized) box parameters are used for regression target, which can be described as follows.

$$r_x = \frac{x_g - x_a}{w_a}, r_y = \frac{y_g - y_a}{h_a}, r_w = \log \frac{w_g}{w_a}, r_h = \log \frac{h_g}{h_a}, \tag{3.16}$$

where (x_a, y_a, w_a, h_a) and (x_g, y_g, w_g, h_g) denotes the four parameters of the anchor and the ground truth bounding box respectively, (r_x, r_y, r_w, r_h) denotes the refined parameters for regression target. As for the recovery phase, we have

$$x_g^* = x_a + r_x^* w_a, y_g^* = h_a + r_y^* h_a, w_g^* = w_a e^{r_w^*}, h_g^* = h_a e^{r_h^*}, \tag{3.17}$$

where $(r_x^*, r_y^*, r_w^*, r_h^*)$ and $(x_g^*, y_g^*, w_g^*, h_g^*)$ denote the predicted refinement parameters and the parameters of the final predicted bounding box respectively.

The reason why the refined parameters are used for training, instead of original box, is that the former one eliminates the location information thus more reasonable.

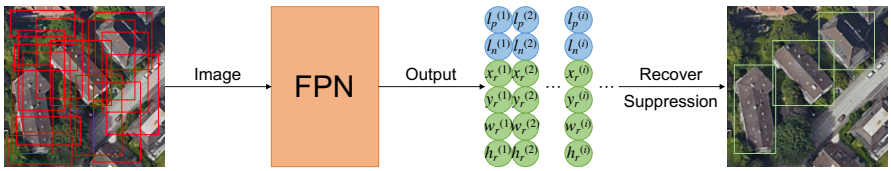


Figure 3.10: Simplified structure of multiple bounding box regression.

Loss As for training, only positive and negative anchors are used, where classification uses both and regression only uses positive anchors. The loss used for classification is the log loss, described as follows.

$$L_c = \mathbb{1}[p = 1] \log \frac{1}{p^*} + \mathbb{1}[p = -1] \log \frac{1}{1 - p^*}, \tag{3.18}$$

where L_c is the classification loss, p is the polarity of the anchor, p' is the predicted probability, $\mathbb{1}[\cdot]$ is the indicator function. The loss used for regression is the smooth L_1 loss, described as follows.

$$f(x) = \begin{cases} \frac{1}{2}x^2, & |x| < 1, \\ |x| - \frac{1}{2}, & |x| \geqslant 1, \end{cases} \tag{3.19}$$

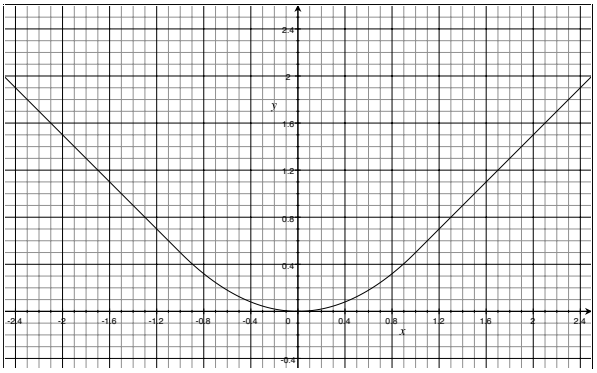


Figure 3.11: Smooth L_1 loss function.

$$L_r = f(x_g^* - x_g) + f(y_g^* - y_g) + f(w_g^* - w_g) + f(h_g^* - h_g), \tag{3.20}$$

where L_r is the regression loss. The total loss is defined as

$$L = \sum_i L_c^{(i)} + \lambda \sum_i \mathbb{1}[p_i = 1] L_r^{(i)}, \tag{3.21}$$

where λ is a self-defined parameter.

3.2.4 Feature Pyramid and Backbone

From equation 3.10 we know that even a small image can have a huge number of possible anchors. However, it is impossible to involve all anchors in the calculation, since the output layer would have too many nodes. Thus, how to select the proper anchors in the original image remains a problem. The selected anchors should cover the ground truth bounding boxes as much as possible. Under this guideline, selected anchors should have different kinds of shapes, in different scales and evenly distributed in the image.

In fact, in RPN from Faster R-CNN, there is a simple way to select anchors. For each entry of the convoluted feature map (or each pixel of the original image with corresponding stride), we generate k different shapes of anchors also in different scales, which is shown in figure 3.12. For example, if we select anchors from image with size 320×320, stride 8×8, 3 different shapes $(1:1, \sqrt{2}/2:\sqrt{2}, \sqrt{2}:\sqrt{2}/2)$ and 4 scales (16, 32, 64, 128), we will have $(320/8)^2 \times 3 \times 4 = 19200$ anchors.

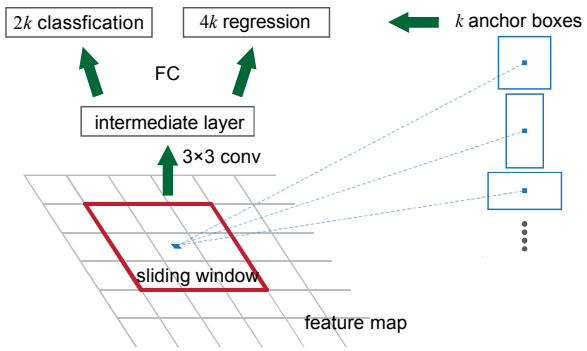


Figure 3.12: Anchor selection.

But this method produces two problems: (1) large anchors are too close to each other, resulting in redundancy; (2) the stride may be too large for small anchors, thus a typically small ground truth bounding box which are sandwiched between two small anchors is very likely to be missed. Both problems are tackled with the introduction of feature pyramid.

Feature pyramid is a basic component in recognition systems for detecting objects. It actually refers to semantic feature maps at different scales. FPN exploits the inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids. The network uses a top-down architecture with lateral connections is developed for building an in-network feature pyramid from a single-scale input. Figure 3.13 shows its architecture.

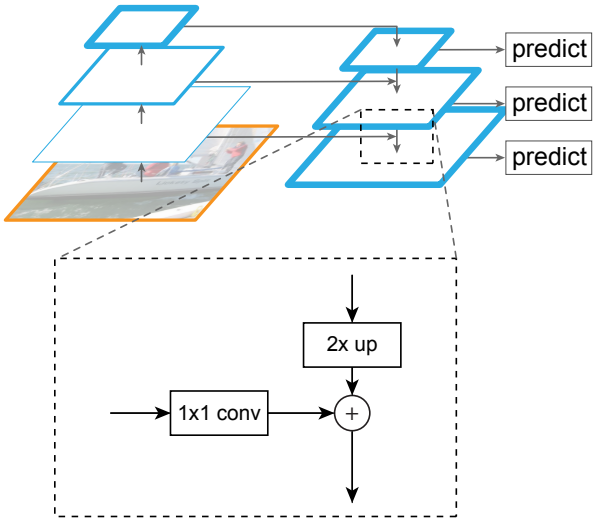


Figure 3.13: Structure of Feature Pyramid Network. The upper left part is the general convolutional process. The upper right part shows the feature pyramid. The lower part shows the lateral connections in detail.

The relationship between the feature pyramid and anchors is that each level of the feature pyramid corresponds to single scale of anchors, which can be seen in figure 3.14. In general, we have

$$S_i = 2^{i-1} S_1, i \in \{1, 2, \ldots\}, \tag{3.22}$$

where S_i denotes the smallest anchor size, S_i denotes the anchor size corresponding to the feature map with 2^{-i} resolution. For example, for an image with size 320×320, if the smallest anchor size is set to be 16, the anchor size for the feature map with resolution 40×40 would be 128.

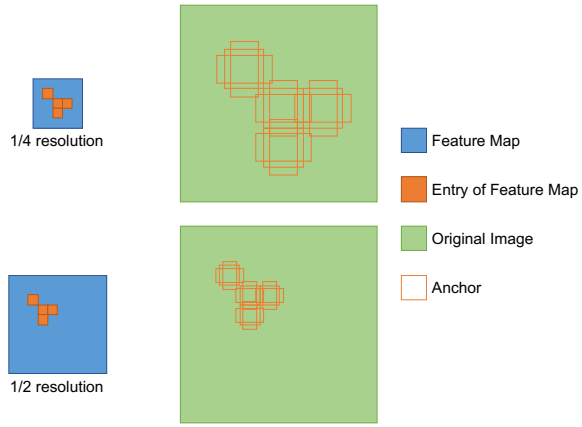


Figure 3.14: Generation of anchors from different layers of feature pyramid.

Therefore, in FPN, the feature map with larger resolution corresponds to

smaller anchor size (i.e. smaller stride with regard to the original image). That is the core reason why the problems mentioned above can be addressed. On the other hand, the total number of anchors n_l has an upper bound $\Theta(wh)$, which can be derived as

$$n_l = \sum_{i=1}^{l} \frac{w}{2^i}\frac{h}{2^i}k = whk\sum_{i=1}^{l} 4^{-i} = \frac{1}{3}whk(1-4^{-l}) \leqslant \frac{1}{3}whk, \tag{3.23}$$

where the subscript l denotes the number of feature maps, i.e. the depth of feature pyramid, w and h denotes the image width and height, k denotes the number of designed anchor shapes. Compared with equation 3.10, the total number of anchors is reduced by two orders of magnitude.

The upper left part of figure 3.13 is so-called backbone of FPN, which is exactly a general CNN. In Mask R-CNN, the backbone used for feature extraction is ResNet-101, which can give excellent gains in both accuracy and speed. However, in our project, VGG-16 is chosen as the backbone of FPN, because we want FPN and PolygonRNN share the same VGG-16. For more details about the shared VGG-16, please refer to subsection 3.3.2.

Figure 3.15 shows the architecture of FPN with VGG-16 backbone used in our project for feature pyramid extraction. Actually, the lateral connections are not fixed. It means that each layer of VGG-16 can be lead out in a reasonable situation. Here we use 4-layer feature pyramid, and features are taken from layers `conv3_3`, `conv4_3` and `conv5_3`. For the network to make further prediction is already shown in figure 3.12.

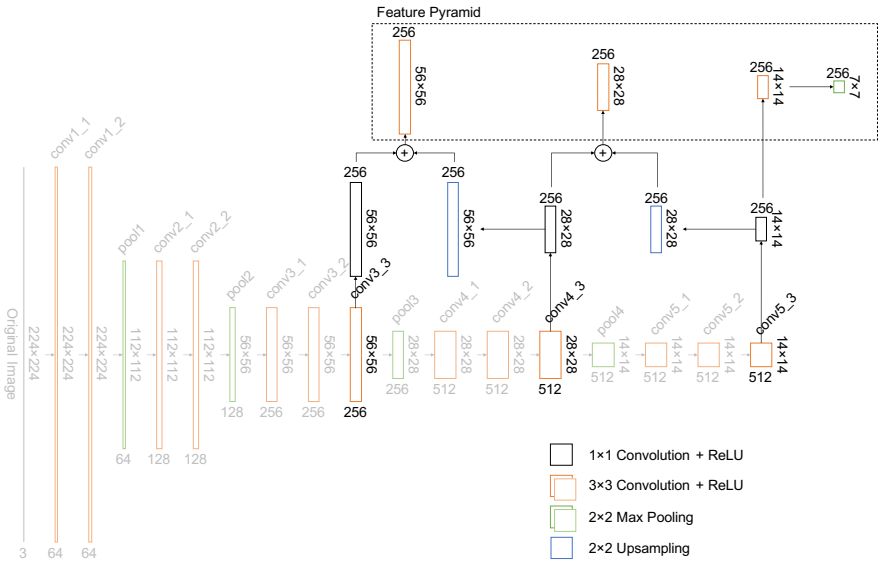


Figure 3.15: FPN with VGG-16 backbone.

3.3 R-PolygonRNN

In this section, the final model, R-PolygonRNN (Region-based PolygonRNN), is derived and introduced in detail. Here we propose 3 possible versions, denoting two-step version, hybrid version and hybrid version with RoIAlign, which are described in subsection 3.3.1, 3.3.2 and 3.3.3, respectively.

3.3.1 Two-step Version

The two-step version is very natural and easy to think of. In the first step, RPN is used to get the RoIs from the aerial image and each RoI contains a building. In the second step, PolygonRNN is used to get the geometrical shape. Thus, our problem is solved after these two steps. Figure 3.16 shows its pipeline.

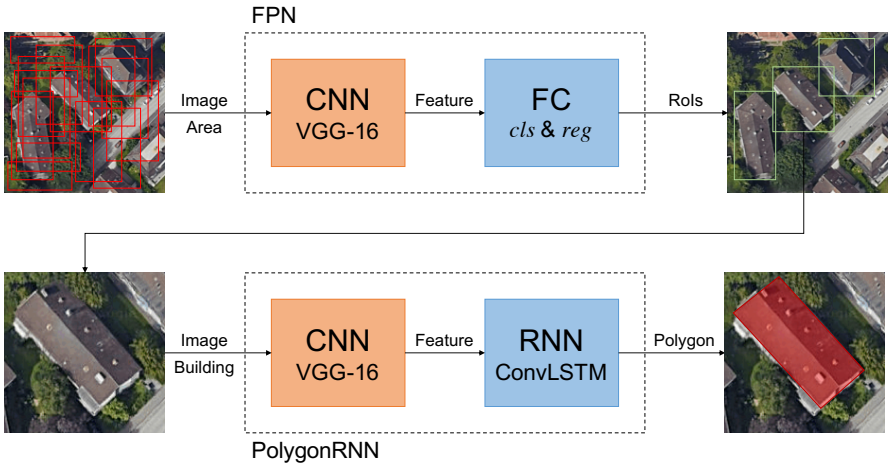


Figure 3.16: R-PolygonRNN , two-step version.

From the figure 3.16 we can see that RPN and PolygonRNN are completely separate. Therefore, during the training phase, we can train the two models separately and integrate them when making prediction.

3.3.2 Hybrid Version

Note that in figure 3.16, either when the area image is inputted to FPN or when the building image is inputted to PolygonRNN, they both pass CNN first, and their CNNs both adopt VGG-16. In fact, regardless of the whole model of FPN or PolygonRNN, the feature extraction in their CNN parts are both from the lateral connection on the basis of VGG-16. Although they extract features from different levels, the backbone of the original VGG-16 is

exactly the same for both of them. Therefore, they can share the same VGG-16 skeleton, just like what figure 3.17 shows.

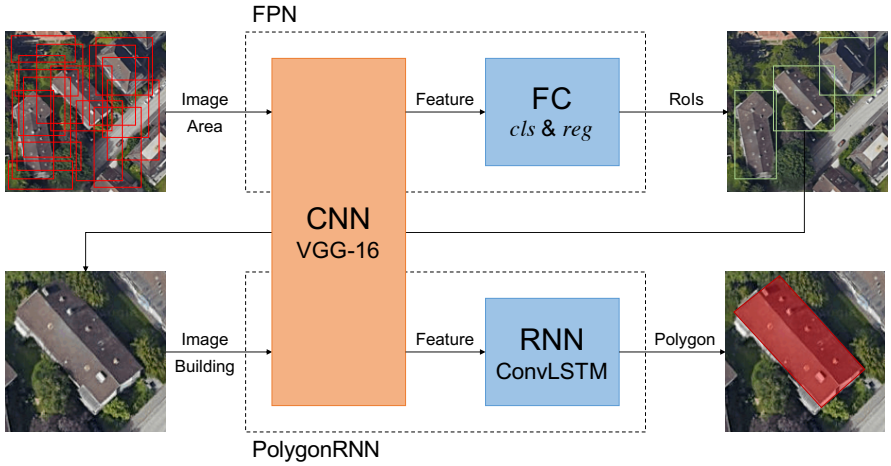


Figure 3.17: R-PolygonRNN , hybrid version.

As a result, the training phase of the whole model, R-PolygonRNN, cannot be separated and multi-task training is needed. In general, multi-tasking training can make the training phase better, and can therefore improve the the model's prediction effect and accuracy.

3.3.3 Hybrid Version with RoIAlign

Note that both two-step version and hybrid version do convolution for the area image and the building image respectively. However, we know that the building image is taken from the area image, so there is redundancy in the twice convolutions. Considering that we already have the feature map of the area image, it is unnecessary to do a second convolution since we can take the building feature from the corresponding position in the feature map of the area image when given the RoI.

In Mask R-CNN, RoIAlign is proposed to solve this kind of problem. Actually its previous version, RoIPooling is already used in Faster R-CNN. The role of RoIPooling is to pool the corresponding area in the feature map into a fixed-size tensor, so as to perform subsequent classification phase. The positions of the RoIs are usually obtained by regression of the model, which are generally floating numbers. RoIPooling has twice process of rounding these floating numbers: (1) box boundaries are rounded to integers; (2) the region of box is divided into several cells, boundaries of each cell are rounded to integers. After the twice rounding process, each cell has a certain deviation from its initial position, which will affect the accuracy of subsequent classi-

fication or segmentation (for Mask R-CNN). This problem is summarized as 'misalignment' in Mask R-CNN paper.

In order to tackle the shortcomings of RoIPooling, Mask R-CNN introduces RoIAlign, using bilinear interpolation instead of rounding numbers. It can be used to obtain the value at the pixel whose coordinates are floating numbers, thereby transforming the pooling process into a continuous operation.

Therefore, we can simply apply the bounding boxes obtained from FPN calculation directly on the feature map by RoIAlign and send the pooled features to the RNN part of PolygonRNN. Figure 3.18 shows R-PolygonRNNmodel architecture with RoIAlign.

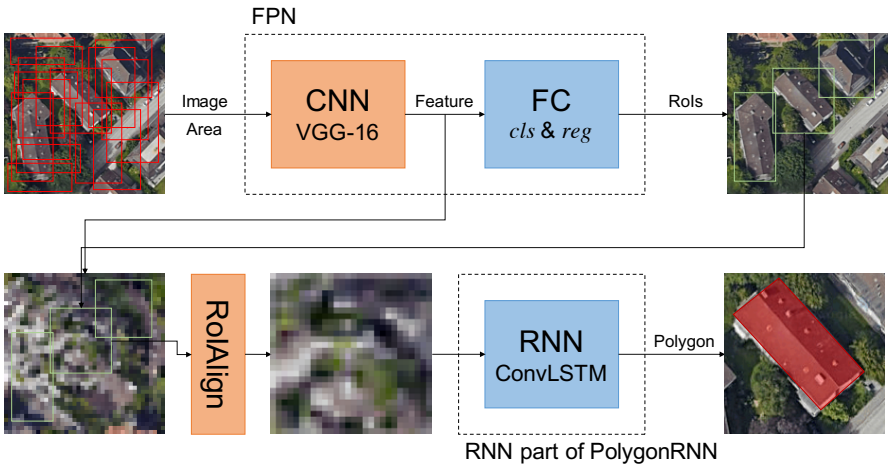


Figure 3.18: R-PolygonRNN , hybrid version with RoIAlign.

As a result, VGG-16 is used only and therefore it can speed up the prediction in theory.

Chapter 4

Experiments and Results

In this chapter, the entire experimental process is presented, from the acquisition of the ground truth dataset (see section 4.1), to the implementation, training and prediction phases of the R-PolygonRNN (see section 4.2), and finally to the results evaluation and analysis (see section 4.3).

4.1 Ground Truth

Our ground truth dataset consists of satellite images and buildings' coordinates, which are used as inputs and labels respectively when training. In this project, all of the satellite images are collected from Google Static Maps API and all of the latitude and longitude coordinates of the polygon vertices of buildings are collected from OpenStreetMap. For details of the two APIs mentioned above, please refer to subsections 4.1.1 and 4.1.2.

As mentioned in section 3.3, the training phase of our model R-PolygonRNN requires two different kinds of ground truth dataset, areas with multiple bounding boxes for FPN and buildings with geometrical shapes for PolygonRNN, all of which are illustrated in subsection 4.1.3.

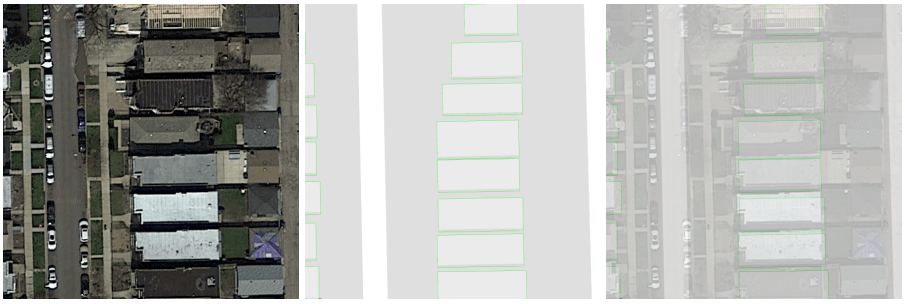
Since the whole dataset is collected from two different sources, the problem of inconsistency may exist. Subsection 4.1.4 describes details of the problems in the ground truth dataset, and subsection 4.1.5 proposes a solution to adjust the shift between buildings' images and polygons.

4.1.1 Google Static Maps API

Google Static Maps API[1] provides an interface that implements maps as high-resolution images. Users can download customized map based on URL with different parameters, which is sent through a standard HTTPS request.

[1] `https://developers.google.com/maps/documentation/static-maps/`

The parameters in URL includes the map type (satellite, roadmap, etc.), latitude and longitude coordinates of the image center, the resolution of the image, the zoom level, and the scale. For the usage of the Google Static Maps API, please refer to section A.1 in appendices. Figure 4.1(a) and 4.1(b) shows two types of images can be obtained by Google Static Maps API.



(a) satellite image (b) roadmap image (c) hybrid image

Figure 4.1: Example images downloaded through Google Static Maps API. (a) and (b) are obtained directly by the API, with 41.9399708 degrees north latitude, 87.7380649 degrees west longitude of the center (located in Chicago), both width and height 600 pixels, zoom level 20 and scale 1, but different in map types. (c) is obtained by overlaying (a) with (b), with 75% opacity. It can be clearly seen that the two images cannot match very well.

We would have liked to obtain buildings' coordinates from Google Static Maps API as well, but we find that the API does not provide such information directly. Instead, it gives the styled roadmap images like figure 4.1(b), where querying coordinates requires further corner detection but the boundaries may be still inaccurate (see figure 4.1(c) for example). Thus, this thesis project, only satellite images from Google Static Maps API are used.

4.1.2 OpenStreetMap

OpenStreetMap[2] provides an interface that implements a map as a XML format `.osm` file. The file contains all the information which can be used to recover the map. When the map's bounding box is given, users can retrieve the latitude and longitude coordinates of the buildings' vertices and roads' central lines within the map. For the usage of OpenStreetMap and the format of the `.osm` file, please refer to section A.2 in appendices.

The coordinates obtained by the API is the original latitude and longitude coordinates of the buildings' vertices, which cannot be directly used for training. We need to convert them to relative pixel coordinates with regards to an given satellite image.

[2] `https://www.openstreetmap.org/`

As a matter of fact, every fixed point on the earth can correspond to a unique pixel on the map at a specific zoom level. This projection process can be regarded as a function. Since we have already known the relative pixel coordinates of the image center (half width and half height), the projection for an arbitrary point with given latitude and longitude coordinates can be therefore computed. This process can be described as following equations.

$$(c_x, c_y) = (\frac{w}{2}, \frac{h}{2}) = f(c_{lon}, c_{lat}, z) + (\Delta_x, \Delta_y), \tag{4.1}$$

$$(p_x, p_y) = f(p_{lon}, p_{lat}, z) + (\Delta_x, \Delta_y), \tag{4.2}$$

where f is the function that projects latitude and longitude coordinates to global pixel coordinates (see A.3 in appendices for more details), c is the image center, p is an arbitrary point, the subscripts x, y, lon, lat mean the relative pixel and longitude and latitude coordinates respectively, z is the zoom level, Δ is the translation constant from the global to the relative pixel coordinate system. Thus, we have

$$(p_x, p_y) = f(p_{lon}, p_{lat}, z) - f(c_{lon}, c_{lat}, z) + (\frac{w}{2}, \frac{h}{2}), \tag{4.3}$$

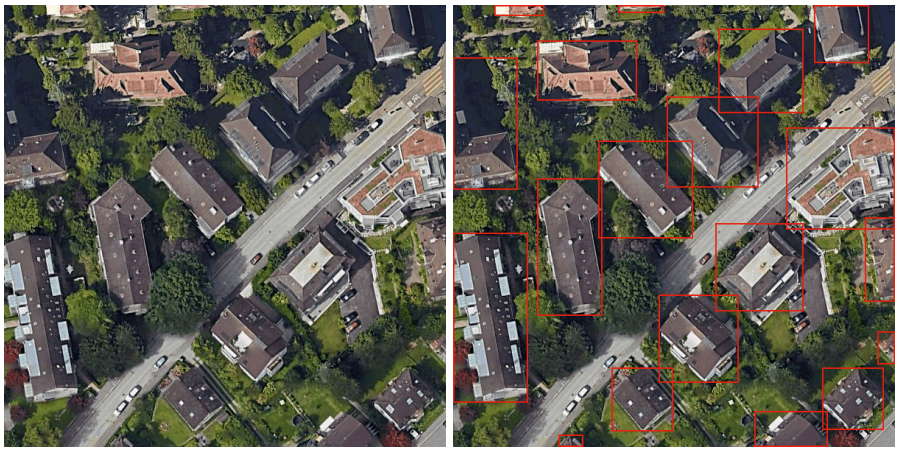
for any arbitrary point p.

4.1.3 Areas and Buildings

As mentioned in section 3.3, R-PolygonRNN is formed by FPN and PolygonRNN. However, the training phase of these two models requires two different kinds of ground truth dataset.

Areas for FPN In subsection 3.2 we have shown that FPN aims at finding rectangular regions of interests. Thus, training FPN generally requires a relatively large image with several objects in it. When considering our problem, an example of the ground truth can be an area of a city with several bounding boxes. Each box contains a single building, regardless of its tight polygon outline. Figure 4.2 gives an example area for training FPN and its visualized ground truth label.

Buildings for PolygonRNN In section 3.1 we have mentioned that PolygonRNN can only deal with images with single object, meaning that the bounding box of the object should be first given. Thus, training PolygonRNN generally requires a relatively small image with single object, as well as the information of its polygon outline. When it comes to our problem, an example of the ground truth can be a building with some padding and the pixel coordinates of building's vertices. Figure 4.3 gives example buildings for training PolygonRNN and its visualized ground truth label.



(a) example area (b) example area with bounding boxes

Figure 4.2: An example area in Zurich for FPN training. (a) is the original satellite image obtained by Google Static Maps API. (b) is (a) covered by multiple bounding boxes of buildings, which is visualized based on the ground truth.

(a) example buildings

(b) example buildings with polygon outlines

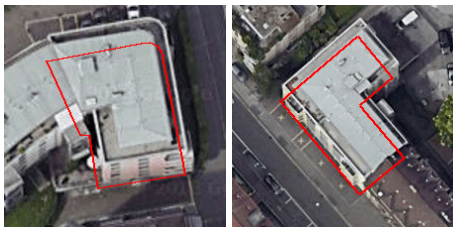
Figure 4.3: Example buildings in Zurich for PolygonRNN training. (a) is the original satellite images obtained by Google Static Maps API. (b) is (a) covered by the corresponding polygons' edges, which are visualized based on the ground truth.

4.1.4 Problems

Recall that the satellite images and polygons' coordinates are collected from two different sources, the problem of inconsistency may exist. That is to say, the polygon and the actual building we see in the image may not match well in some cases. This inconsistency is mainly reflected in the following aspects.

(a) general shift

(b) polygon size and shape



(c) angle issue

(d) adjacent buildings

Figure 4.4: Problems existed in the ground truth dataset.

General Shift Figure 4.4(a) gives two typical shift examples in the dataset. This shift is generally within the range of 30 pixels, either up and down, or left and right.

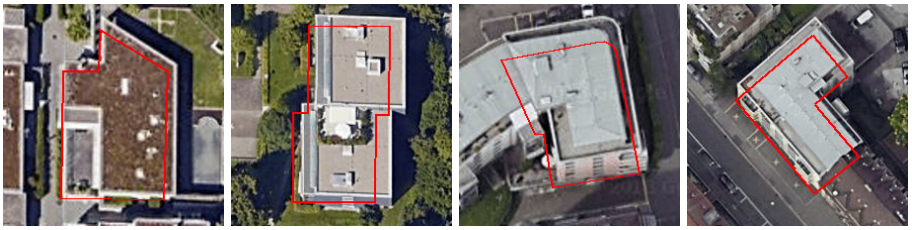
Polygon Size and Shape Sometimes the polygon size and shape is different from the actual building size as well. This is typically because OpenStreetMap measures the bottom of the building and the aerial image is taken from the top. The roof of the house we see from the top can be different from the floor area. Figure 4.4(b) shows a building with a smaller polygon and a building with a too rough outline.

Angle Issue Some tall buildings may suffer from the angle issue. This is because the satellite cannot always be vertically right above the building, so the side of these very high buildings will be photographed. Figure 4.4(c) gives two examples.

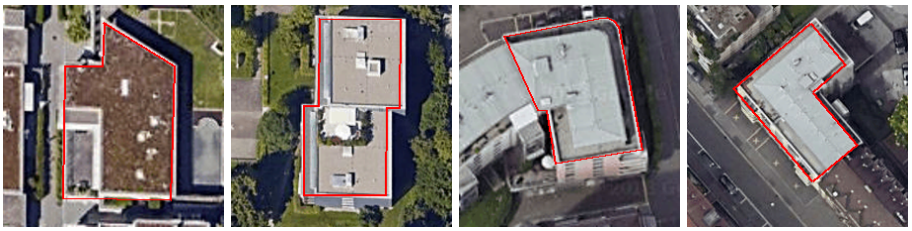
Adjacent Buildings Some adjacent buildings will be very close together, and some even share a common roof. Thus, from the satellite image, it is generally difficult to distinguish the boundaries between them. However, these buildings are separated in the dataset of OpenStreetMap. It means that where there are originally edges between buildings, it looks like there is no edge in the image. This kind of inaccuracy would have negative effect on the training phase. Figure 4.4(d) and gives two examples.

4.1.5 Adjustments

Problems such as too rough polygon shape, angle issue and adjacent buildings are unsolvable unless the data sources can correct themselves. What we can improve is to tackle the general shift and the polygon size problem and make the image-polygon pairs more matching. By observation, we can get to some conclusions for a matching image-polygon pair.



(a) before adjustment



(b) after adjustment

Figure 4.5: Adjustment examples. (a) shows the polygons before adjustment, while (b) shows the polygons after adjustment.

Color Variance The color variance of image pixels covered by polygon generally reaches the local minimum, because the roof of a building usually has only one color or several similar colors.

Edges The polygon edges generally lie within the output of the edge detection (e.g. Sobel, Canny) performed on the image.

Corners The polygon vertices can generally match the output of the corner detection (e.g. Harris, SIFT) performed on the image.

Based on these conclusions, we propose a brute force shift adjustment method. Briefly, we resize and translate the initial polygon within a certain range, and traverse all the cases to see where the minimum color variance, maximum edge and corner coverage are reached. Figure 4.5 illustrates some examples which compare the polygons before and after the adjustment. Results show that this algorithm can well address the shift problem to some extent.

4.2 Implementation Details

4.2.1 Configuration

number of layers LSTM polygon anticlockwise remove point in a line

4.2.2 Training

Furthermore, buildings around the edges of areas are typically ignored

more likely to be incomplete. In this case, the polygon would be clipped by the edges, resulting in new vertices, which can be very hard to compute.

(*figureplaceholder*)

Although buildings can extract from areas

Therefore, in this project, two different kinds of ground truth dataset are collected separately. batchsizeBuildingareaPolygonRNNbatchsize

4.2.3 Prediction

Beam search

4.3 Experiment Results

Dummy text.

4.3.1 Single Building Segmentation

Dummy text.

4.3.2 Buildings Localization

Dummy text.

4.3.3 R-PolygonRNN

Beam search

Example Subsubsection

Dummy text.

Example Paragraph Dummy text.

Example Subparagraph Dummy text.

Chapter 5

Problems and Future Work

Dummy text.

5.1 Problems

5.1.1 Problem 1

Dummy text.

5.1.2 Problem 2

aaaaa

5.2 Future Work

Dummy text.

5.2.1 Ground Truth

GTbbox4.1.3

5.2.2 Training Phase

alternative

5.2.3 Model Generalization

Region-based PolygonRNN

Chapter 6

Introduction

This is version `v1.4` of the template.

We assume that you found this template on our institute's website, so we do not repeat everything stated there. Consult the website again for pointers to further reading about LaTeX. This chapter only gives a brief overview of the files you are looking at.

6.1 Features

The rest of this document shows off a few features of the template files. Look at the source code to see which macros we used!

The template is divided into TeX files as follows:

1. `thesis.tex` is the main file.
2. `extrapackages.tex` holds extra package includes.
3. `layoutsetup.tex` defines the style used in this document.
4. `theoremsetup.tex` declares the theorem-like environments.
5. `macrosetup.tex` defines extra macros that you may find useful.
6. `introduction.tex` contains this text.
7. `sections.tex` is a quick demo of each sectioning level available.
8. `refs.bib` is an example bibliography file. You can use BibTeX to quote references. For example, read [**?**] if you can get a hold of it.

6.1.1 Extra package includes

The file `extrapackages.tex` lists some packages that usually come in handy. Simply have a look at the source code. We have added the following comments based on our experiences:

REC This package is recommended.

OPT This package is optional. It usually solves a specific problem in a clever way.

ADV This package is for the advanced user, but solves a problem frequent enough that we mention it. Consult the package's documentation.

As a small example, here is a reference to the Section *Features* typeset with the recommended *varioref* package:

> See Section 6.1 on the preceding page.

6.1.2 Layout setup

This defines the overall look of the document – for example, it changes the chapter and section heading appearance. We consider this a 'do not touch' area. Take a look at the excellent *Memoir* documentation before changing it.

In fact, take a look at the excellent *Memoir* documentation, full stop.

6.1.3 Theorem setup

This file defines a bunch of theorem-like environments.

Theorem 6.1 *An example theorem.*

Proof Proof text goes here. □

Note that the q.e.d. symbol moves to the correct place automatically if you end the proof with an `enumerate` or `displaymath`. You do not need to use `\qedhere` as with *amsthm*.

Theorem 6.2 (Some Famous Guy) *Another example theorem.*

Proof This proof

1. ends in an enumerate. □

Proposition 6.3 *Note that all theorem-like environments are by default numbered on the same counter.*

Proof This proof ends in a display like so:

$$f(x) = x^2. \qquad \square$$

6.1.4 Macro setup

For now the macro setup only shows how to define some basic macros, and how to use a neat feature of the *mathtools* package:

$$|a|, \quad \left|\frac{a}{b}\right|, \quad |\frac{a}{b}|.$$

Chapter 7

Writing scientific texts in English

This chapter was originally a separate document written by Reto Sphel. It is reprinted here so that the template can serve as a quick guide to thesis writing, and to provide some more example material to give you a feeling for good typesetting.

7.1 Basic writing rules

The following rules need little further explanation; they are best understood by looking at the example in the booklet by Knuth et al., 2–3.

Rule 7.1 Write texts, not chains of formulas.

More specifically, write full sentences that are logically interconnected by phrases like 'Therefore', 'However', 'On the other hand', etc. where appropriate.

Rule 7.2 Displayed formulas should be embedded in your text and punctuated with it.

In other words, your writing should not be divided into 'text parts' and 'formula parts'; instead the formulas should be tied together by your prose such that there is a natural flow to your writing.

7.2 Being nice to the reader

Try to write your text in such a way that a reader enjoys reading it. That's of course a lofty goal, but nevertheless one you should aspire to!

Rule 7.3 Be nice to the reader.

Give some intuition or easy example for definitions and theorems which might be hard to digest. Remind the reader of notations you introduced

many pages ago – chances are he has forgotten them. Illustrate your writing with diagrams and pictures where this helps the reader. Etc.

Rule 7.4 Organize your writing.

Think carefully about how you subdivide your thesis into chapters, sections, and possibly subsections. Give overviews at the beginning of your thesis and of each chapter, so the reader knows what to expect. In proofs, outline the main ideas before going into technical details. Give the reader the opportunity to 'catch up with you' by summing up your findings periodically.

Useful phrases: 'So far we have shown that ...', 'It remains to show that ...', 'Recall that we want to prove inequality (7), as this will allow us to deduce that ...', 'Thus we can conclude that .... Next, we would like to find out whether ...', etc.

Rule 7.5 Don't say the same thing twice without telling the reader that you are saying it twice.

Repetition of key ideas is important and helpful. However, if you present the same idea, definition or observation twice (in the same or different words) without telling the reader, he will be looking for something new where there is nothing new.

Useful phrases: 'Recall that [we have seen in Chapter 5 that] ...', 'As argued before / in the proof of Lemma 3, ...', 'As mentioned in the introduction, ...', 'In other words, ...', etc.

Rule 7.6 Don't make statements that you will justify later without telling the reader that you will justify them later.

This rule also applies when the justification is coming right in the next sentence! The reasoning should be clear: if you violate it, the reader will lose valuable time trying to figure out on his own what you were going to explain to him anyway.

Useful phrases: 'Next we argue that ...', 'As we shall see, ...', 'We will see in the next section that ..., etc.

7.3 A few important grammar rules

Rule 7.7 There is (almost) *never* a comma before 'that'.

It's really that simple. Examples:

> We assume that ...
> *Wir nehmen an, dass ...*

It follows that ...
Daraus folgt, dass ...

'thrice' is a word that is seldom used.
'thrice' ist ein Wort, das selten verwendet wird.

Exceptions to this rule are rare and usually pretty obvious. For example, you may end up with a comma before 'that' because 'i.e.' is spelled out as 'that is':

For $p(n) = \log n/n$ we have ... However, if we choose p a little bit higher, that is $p(n) = (1+\varepsilon)\log n/n$ for some $\varepsilon > 0$, we obtain that...

Or you may get a comma before 'that' because there is some additional information inserted in the middle of your sentence:

Thus we found a number, namely n_0, that satisfies equation (13).

If the additional information is left out, the sentence has no comma:

Thus we found a number that satisfies equation (13).

(For 'that' as a relative pronoun, see also Rules 7.9 and 7.10 below.)

Rule 7.8 There is usually no comma before 'if'.

Example:

A graph is not 3-colorable if it contains a 4-clique.
Ein Graph ist nicht 3*-frbbar, wenn er eine* 4*-Clique enthlt.*

However, if the 'if' clause comes first, it is usually separated from the main clause by a comma:

If a graph contains a 4-clique, it is not 3-colorable .
Wenn ein Graph eine 4*-Clique enthlt, ist er nicht* 3*-frbbar.*

There are more exceptions to these rules than to Rule 7.7, which is why we are not discussing them here. Just keep in mind: don't put a comma before 'if' without good reason.

Rule 7.9 Non-defining relative clauses have commas.

Rule 7.10 Defining relative clauses have no commas.

In English, it is very important to distinguish between two types of relative clauses: defining and non-defining ones. This is a distinction you absolutely need to understand to write scientific texts, because mistakes in this area actually distort the meaning of your text!

It's probably easier to explain first what a *non-defining* relative clause is. A non-defining relative clauses simply gives additional information *that could also be left out* (or given in a separate sentence). For example, the sentence

> The WEIRDSORT algorithm, which was found by the famous mathematician John Doe, is theoretically best possible but difficult to implement in practice.

would be fully understandable if the relative clause were left out completely. It could also be rephrased as two separate sentences:

> The WEIRDSORT algorithm is theoretically best possible but difficult to implement in practice. [By the way,] WEIRDSORT was found by the famous mathematician John Doe.

This is what a non-defining relative clause is. *Non-defining relative clauses are always written with commas.* As a corollary we obtain that you cannot use 'that' in non-defining relative clauses (see Rule 7.7!). It would be wrong to write

> ~~The WEIRDSORT algorithm, that was found by the famous mathematician John Doe, is theoretically best possible but difficult to implement in practice.~~

A special case that warrants its own example is when 'which' is referring to the entire preceding sentence:

> Thus inequality (7) is true, which implies that the Riemann hypothesis holds.

As before, this is a non-defining relative sentence (it could be left out) and therefore needs a comma.

So let's discuss *defining* relative clauses next. A defining relative clause tells the reader *which specific item the main clause is talking about*. Leaving it out either changes the meaning of the sentence or renders it incomprehensible altogether. Consider the following example:

> The WEIRDSORT algorithm is difficult to implement in practice. In contrast, the algorithm that we suggest is very simple.

Here the relative clause 'that we suggest' cannot be left out – the remaining sentence would make no sense since the reader would not know which algorithm it is talking about. This is what a defining relative clause is. *Defining relative clauses are never written with commas.* Usually, you can use both 'that' and 'which' in defining relative clauses, although in many cases 'that' sounds better.

As a final example, consider the following sentence:

> For the elements in $\mathcal{B}$ which satisfy property (A), we know that equation (37) holds.

Table 7.1: Things you (usually) don't say

~~It holds (that) ...~~	We have ...	*Es gilt ...*
('Equation (5) holds.' is fine, though.)		
~~*x* fulfills property $\mathcal{P}$.~~	x satisfies property $\mathcal{P}$.	*x erfllt Eigenschaft $\mathcal{P}$.*
~~in average~~	on average	*im Durchschnitt*
~~estimation~~	estimate	*Abschtzung*
~~composed number~~	composite number	*zusammengesetzte Zahl*
~~with the help of~~	using	*mit Hilfe von*
~~surely~~	clearly	*sicher, bestimmt*
~~monotonously increasing~~	monotonically incr.	*monoton steigend*
(Actually, in most cases 'increasing' is just fine.)		

This sentence does not make a statement about all elements in $\mathcal{B}$, only about those satisfying property (A). The relative clause is *defining*. (Thus we could also use 'that' in place of 'which'.)

In contrast, if we add a comma the sentence reads

> For the elements in $\mathcal{B}$, which satisfy property (A), we know that equation (37) holds.

Now the relative clause is *non-defining* – it just mentions in passing that all elements in $\mathcal{B}$ satisfy property (A). The main clause states that equation (37) holds for *all* elements in $\mathcal{B}$. See the difference?

7.4 Things you (usually) don't say in English – and what to say instead

Table 7.1 lists some common mistakes and alternatives. The entries should not be taken as gospel – they don't necessarily mean that a given word or formulation is wrong under all circumstances (obviously, this depends a lot on the context). However, in nine out of ten instances the suggested alternative is the better word to use.

Chapter 8

Typography

8.1 Punctuation

Rule 8.1 Use opening (‘) and closing (’) quotation marks correctly.

In LaTeX, the closing quotation mark is typed like a normal apostrophe, while the opening quotation mark is typed using the French *accent grave* on your keyboard (the *accent grave* is the one going down, as in *frre*).

Note that any punctuation that *semantically* follows quoted speech goes inside the quotes in American English, but outside in Britain. Also, Americans use double quotes first. Oppose

> “Using ‘lasers,’ we punch a hole in . . . the Ozone Layer,” Dr. Evil said.

to

> ‘Using “lasers”, we punch a hole in . . . the Ozone Layer’, Dr. Evil said.

Rule 8.2 Use hyphens (-), en-dashes (–) and em-dashes (—) correctly.

A hyphen is only used in words like ‘well-known’, ‘3-colorable’ etc., or to separate words that continue in the next line (which is known as hyphenation). It is entered as a single ASCII hyphen character (-).

To denote ranges of numbers, chapters, etc., use an en-dash (entered as two ASCII hyphens --) with no spaces on either side. For example, using Equations (1)–(3), we see. . .

As the equivalent of the German *Gedankenstrich*, use an en-dash with spaces on both sides – in the title of Section 7.4, it would be wrong to use a hyphen instead of the dash. (Some English authors use the even longer emdash (—)

instead, which is typed as three subsequent hyphens in LaTeX. This emdash is used without spaces around it—like so.)

8.2 Spacing

Rule 8.3 Do not add spacing manually.

You should never use the commands `\\` (except within tabulars and arrays), `\␣` (except to prevent a sentence-ending space after Dr. and such), `\vspace`, `\hspace`, etc. The choices programmed into LaTeX and this style should cover almost all cases. Doing it manually quickly leads to inconsistent spacing, which looks terrible. Note that this list of commands is by no means conclusive.

Rule 8.4 Judiciously insert spacing in maths where it helps.

This directly contradicts Rule 8.3, but in some cases TeX fails to correctly decide how much spacing is required. For example, consider

$$f(a,b) = f(a+b,a-b).$$

In such cases, inserting a thin math space `\,` greatly increases readability:

$$f(a,b) = f(a+b,\, a-b).$$

Along similar lines, there are variations of some symbols with different spacing. For example, Lagrange's Theorem states that $|G| = [G : H]|H|$, but the proof uses a bijection $f\colon aH \to bH$. (Note how the first colon is symmetrically spaced, but the second is not.)

Rule 8.5 Learn when to use `\␣` and `\@`.

Unless you use 'french spacing', the space at the end of a sentence is slightly larger than the normal interword space.

The rule used by TeX is that any space following a period, exclamation mark or question mark is sentence-ending, except for periods preceded by an uppercase letter. Inserting `\` before a space turns it into an interword space, and inserting `\@` before a period makes it sentence-ending. This means you should write

```
Prof.\ Dr.\ A. Steger is a member of CADMO\@.
If you want to write a thesis with her, you
should use this template.
```

which turns into

> Prof. Dr. A. Steger is a member of CADMO. If you want to write a thesis with her, you should use this template.

The effect becomes more dramatic in lines that are stretched slightly during justification:

> Prof. Dr. A. Steger is a member of CADMO. If you

Rule 8.6 Place a non-breaking space (~) right before references.

This is actually a slight simplification of the real rule, which should invoke common sense. Place non-breaking spaces where a line break would look 'funny' because it occurs right in the middle of a construction, especially between a reference type (Chapter) and its number.

8.3 Choice of 'fonts'

Professional typography distinguishes many font attributes, such as family, size, shape, and weight. The choice for sectional divisions and layout elements has been made, but you will still occasionally want to switch to something else to get the reader's attention. The most important rule is very simple.

Rule 8.7 When emphasising a short bit of text, use `\emph`.

In particular, *never* use bold text (`\textbf`). Italics (or Roman type if used within italics) avoids distracting the eye with the huge blobs of ink in the middle of the text that bold text so quickly introduces.

Occasionally you will need more notation, for example, a consistent typeface used to identify algorithms.

Rule 8.8 Vary one attribute at a time.

For example, for WEIRDSORT we only changed the shape to small caps. Changing two attributes, say, to bold small caps would be excessive (LaTeX does not even have this particular variation). The same holds for mathematical notation: the reader can easily distinguish g_n, $G(x)$, $\mathcal{G}$ and G.

Rule 8.9 Never underline or uppercase.

No exceptions to this one, unless you are writing your thesis on a typewriter. Manually. Uphill both ways. In a blizzard.

8.4 Displayed equations

Rule 8.10 Insert paragraph breaks *after* displays only where they belong. Never insert paragraph breaks *before* displays.

LaTeX translates sequences of more than one linebreak (i.e., what looks like an empty line in the source code) into a paragraph break in almost all contexts. This also happens before and after displays, where extra spacing is inserted to give a visual indication of the structure. Adding a blank line in these places may look nice in the sources, but compare the resulting display

$$a = b$$

to the following:

$$a = b$$

The first display is surrounded by blank lines, but the second is not. It is bad style to start a paragraph with a display (you should always tell the reader what the display means first), so the rule follows.

Rule 8.11 Never use `eqnarray`.

It is at the root of most ill-spaced multiline displays. The *amsmath* package provides better alternatives, such as the `align` family

$$\begin{aligned} f(x) &= \sin x, \\ g(x) &= \cos x, \end{aligned}$$

and `multline` which copes with excessively long equations:

$$\begin{aligned} &\mathrm{P}\bigl[X_{t_0} \in (z_0, z_0 + dz_0], \ldots, X_{t_n} \in (z_n, z_n + dz_n]\bigr] \\ &\qquad = \nu(dz_0) K_{t_1}(z_0, dz_1) K_{t_2 - t_1}(z_1, dz_2) \cdots K_{t_n - t_{n-1}}(z_{n-1}, dz_n). \end{aligned}$$

8.5 Floats

By default this style provides floating environments for tables and figures. The general structure should be as follows:

```
\begin{figure}
  \centering
  % content goes here
  \caption{A short caption}
  \label{some-short-label}
\end{figure}
```

Note that the label must follow the caption, otherwise the label will refer to the surrounding section instead. Also note that figures should be captioned at the bottom, and tables at the top.

The whole point of floats is that they, well, *float* to a place where they fit without interrupting the text body. This is a frequent source of confusion and changes; please leave it as is.

Rule 8.12 Do not restrict float movement to only 'here' (`h`).

If you are still tempted, you should avoid the float altogether and just show the figure or table inline, similar to a displayed equation.

Appendix A

Appendix

A.1 Example URL of Google Static Maps API

`https://maps.googleapis.com/maps/api/staticmap?maptype=satellite&center=4`

`&style=feature:all|element:labels|visibility:off&style=feature:landscape.`

A.2 OpenStreetMap

According to the Google Maps JavaScript API [1],

A.3 Projection

A.4 Anchor Assignment

$$A_{ij} = \begin{cases} 1, & \text{if } p_i = 1 \text{ and } j = d_i, \text{ or} \\ & j \notin \{d_i \mid i = 1, 2, \ldots\} \text{ and } i = \underset{k}{\operatorname{argmax}} \operatorname{IoU}(a_k, g_j) \\ 0, & \text{otherwise,} \end{cases} \tag{A.1}$$

$$A_{ij} = \begin{cases} 1, & \text{if } a_i \text{ is assigned to } g_j, \\ 0, & \text{otherwise,} \end{cases} \tag{A.2}$$

where A is the assignment matrix.

A.5 Non-max Suppression

tms hfm shi bai chi.

[1] https://developers.google.com/maps/documentation/javascript/coordinates

List of Figures

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.