Master Thesis

Building Instance Segmentation and Building Type Detection

Author:
Yaxiong Luo

Supervisor:
Prof. Alexandre Alahi
Assistants:
Saeed Saadatnejad
Alireza Khodaverdian

École Polytechnique Fédérale de Lausanne
Electrical and Electronic Section
Lausanne, 2021.07.09

Abstract

In this thesis, we tackle the task of building instance segmentation and building type detection by using the street view image in Basel. We present 2 models to predict them.

Model 1 is a cascaded model. It first uses the fined-tunned Mask RCNN model pre-trained on the Cityscapes dataset, to predict the building instance and opening (door and window), then determines the building type by the ratio of the opening area in the building. Model 2 is a multi-task model, it trains a new fined-tunned Mask RCNN model to predict the instance and type.

Since one single building in the street can be captured into many images by the camera frame, it will generate duplicated buildings. Building duplication problem is handled in these two ways. The first one is to check if the buildings' global world coordinates are close. The second method is image clustering. For image clustering, DoG-AffNet descriptor is used to doing keypoint feature matching, then DBSCAN clustering is applied to group the buildings by the number of matched keypoint to find the duplication. Finally, we predict the building number and building type ratio in the testing area of Basel.

Keywords: Instance segmentation, Duplication detection, Image Clustering, Mask RCNN, DBSCAN, DoG-AffNet

Acknowledgements

I am very appreciative that *Visual Intelligence for Transportation lab (VITA)* provide me the opportunity to discover the topic relevant with deep learning, image segmentation, and image clustering.

I would like to explain my gratitude to the following people who give me great support and help during my master thesis:

Alireza Khodaverdian and **Saeed Saadatnejad** are my supervisior. During the semester, they are always available to answer my questions, comment about my research method and idea, and review my weekly report. I could not successfully complete my thesis without their kind guidance and motivating participation.

Prof. Alexandre Alahi offers me comprehensive feedback and comment throughout the project. He always wants me to follow the keep up with the state-of-art methods.

Annalisa Casciato, EPFL master student, spent a lot of time annotating the building type and building instances of the training and validation image dataset.

Finally, I would like to thank my parents, who always support me silently, encourage me, protect me, and give all their love to me. They always care about my daily life and teach me lots of life experiences.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the rapid development in artificial intelligence and deep neural networks, most parts of image-based detection problems in the real world are finished by designing a deep neural network. Following this trend, one of the main breakthroughs is making these networks or models detect the object behaving like what human beings see. Because of the depth and width of this problem, the research scholar prefers to divide it into smaller sub-problems. Sub-problems contain image classification, object detection, semantic segmentation, image duplication detection, instance segmentation.

In this thesis, the research aims to develop a method of counting the building number and detect building type in the street of Switzerland's cities from the street view images. This project is related to the topic "instance segmentation", Instance Segmentation is identifying each different object instance of the same class or category within an image. Instance segmentation gives a label to each pixel of the image, so it is a pixel-wise model. Figure 1.1 is a sample example. The dogs in the input image are distinguished into three distinct dogs.

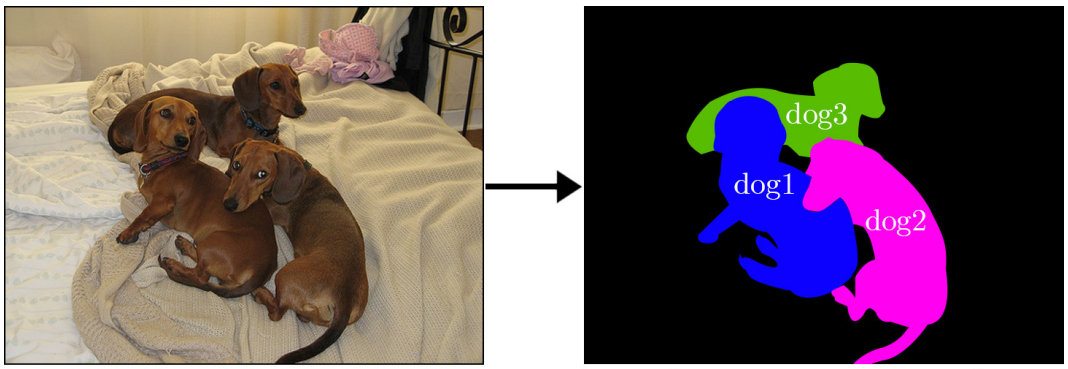


FIGURE 1.1: Instance Segmentation Example

This project can help the local government understand the building type distribution and quantity distribution of buildings and more comprehensively grasp each house's floor distribution and floor height. When the city faces various emergency disasters, the government can effectively use them to carry out risk management and control

quickly since the government has obtained these key data. Also, this can help the government to carry out urban construction and building management.

1.1 Related work

The concept of instance segmentation was firstly published by Hariharan et al in the paper "Simultaneous Detection and Segmentation" [1] in 2014. This topic has been widely progressed by different subtle and gorgeous ideas. A popular and common way is to use some kind of convolutional neural networks as the backbone, then combined with new structures, for instance, recurrent neural networks(RNN) [2], or conditional random fields(CRF) [3].

During the past decade, The performance of the segmentation task has been rapidly improved. These improvements have been driven by powerful ideas, for example, Fast RCNN [4] ,Faster RCNN [5] and Fully Convolutional Network (FCN) [6] frameworks for object detection and semantic segmentation, respectively. In 2017, He et al proposed Mask Region-based Convolution Neural Network(Mask RCNN) [7]. Mask RCNN is one of the popular and effective models to handle this task. After that, there are several new ideas presented [8], [9]. As for the research on the building dataset. JianKang uses a pre-trained CNN model to do the building instance classification on the Google StreetView images dataset [10].

The other task in this thesis is image clustering. It is an unsupervised learning task. The typical methods are K-means [11], spectral clustering [12], hierarchical clustering [13], density-based spatial clustering (DBSCAN) [14]. They can be distinguished into three categories, Partitioning, Hierarchical, and Density-based. we decide to use DBSCAN to do the image clustering since it does not need to know the number of clusters.

Mask RCNN

Mask RCNN is a popular framework for instance segmentation which achieves simultaneous detection, classification and objects segmentation in images.

Mask RCNN extends the target detection framework of Faster R-CNN by adding an extra branch at the end of the model, thus achieving instance segmentation for each output proposal box using a fully connected layer(FC). The segmentation is in parallel with the identification and localization tasks.

The Mask R-CNN framework can be divide into three stages [15](Figure 1.2).

1. The backbone network of Mask R-CNN framework is usually a typical convolutional neural network (ResNet-50/101), the initial layers will detect low-level features with high resolution, and the later layers get features with high semantic accuracy. Then, Feature Pyramid Network (FPN) passes the semantic information contained

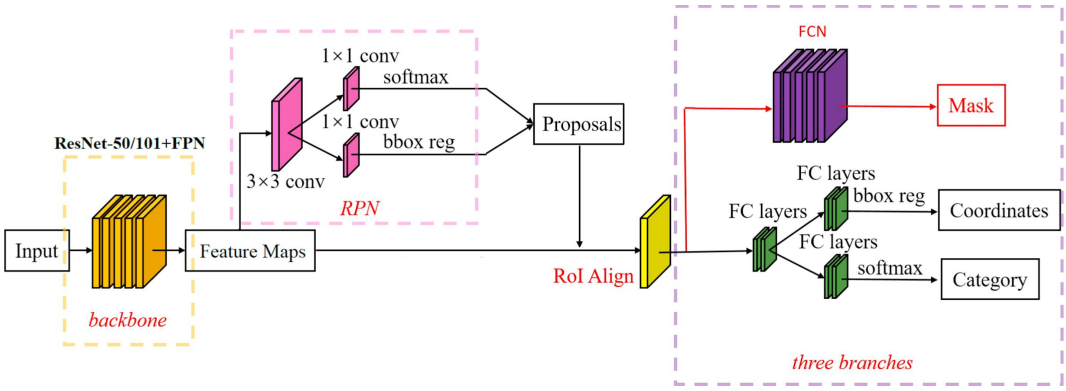


FIGURE 1.2: Mask RCNN Framework

in the high layers to the lower layers with high resolution to accurately detect the small objects. In this stage, it extracted feature maps from input images.

2. Feature maps gotten from the first stage were sent to the region proposal network (RPN). Region Proposal Networks(RPN) uses a sliding window to scan the whole input feature map and generate proposals of regions of interests (ROIs)

3. ROI align head outputted from RPN were mapped to extract the corresponding target features in the shared feature maps. Subsequently, output to the Fully Connected (FC) layers and the fully convolutional network (FCN), respectively. This stage is divided into three branches. The first branch uses FCN to do the building instance segmentation and then gets the building mask. The second branch uses FC layers and regression to get the bounding box coordinate. The third branch uses FC layers to classify to get the categories.

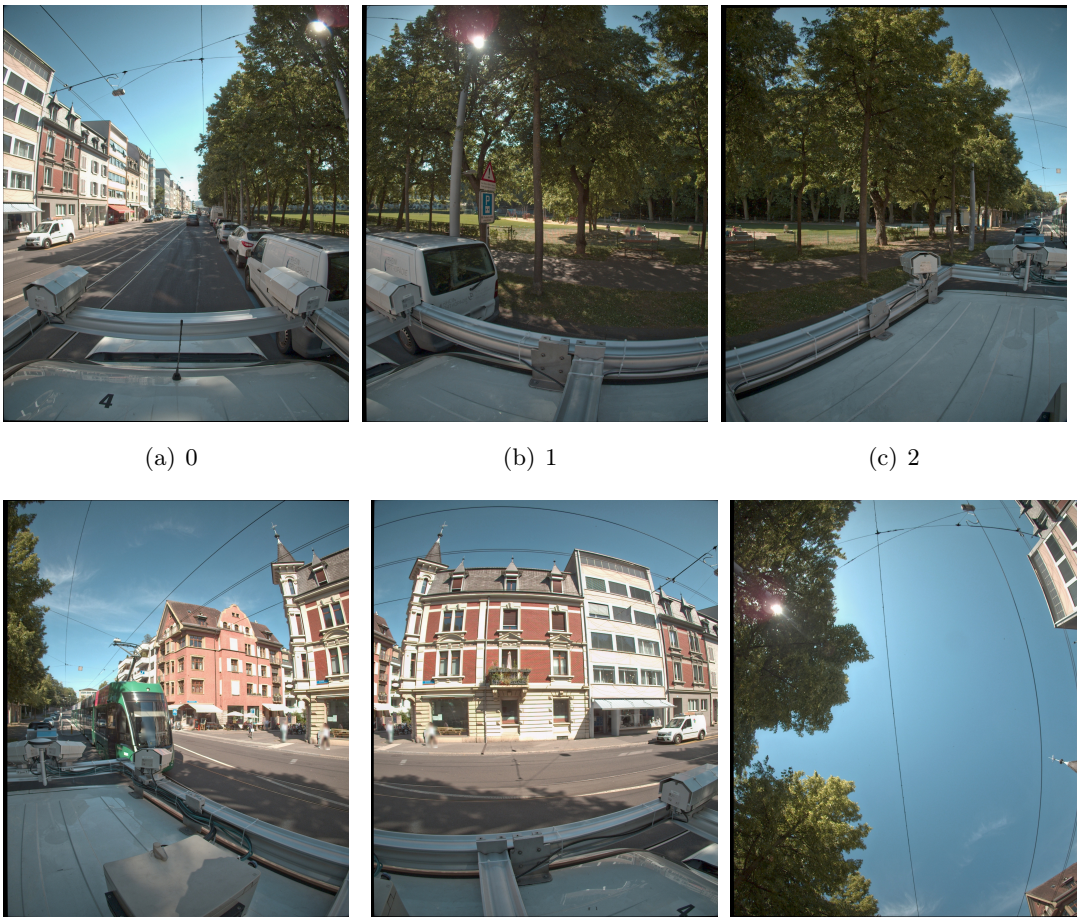
1.2 Outline

In the first chapter, we simply explain the problem we need to tackle and the motivation and objective of this project. Two datasets used in this project are introduced in chapter 2. The overall approach of the project is introduced in chapter 3. In chapter 4 and chapter 5, we explore the cascaded model and the multitask model to detect the building type and building instance. The detailed explanations of the used dataset, architecture, implementation and evaluation are well documented. After getting the building instance and building type, DoG- Affnet feature matching and DBSCAN clustering are used in Chapter 6 to count the building number in the street. In the last chapter, some potential future work and conclusions are presented.

Chapter 2

Datasets

2.1 Basel dataset



(a) 0 (b) 1 (c) 2

(d) 3 (e) 4 (f) 5

FIGURE 2.1: Image taken from panorama camera

A cooperative company provides the Basel dataset. The locations of images in the dataset are in certain streets of Basel downtown, and these images are taken from three types of cameras: stereo, mono, and panorama cameras, but in this project, we only concern about the panorama camera.

Panorama System: A car placed a 360-degree camera (panorama camera) on the roof of the car. The car took pictures along the street and took six photos in six directions for each location. An example is shown in figure 2.1

2.1.1 Data structure and file description

In principle, a street («edge») can have several passages(«Segments») and each passage gets its own video sequence for each sensor(«Stream»). In the video sequence, there are several shooting locations in succession(«Frames»), which in ascending order from zero «0» implicitly result in the recording sequence.

The image dataset follows the standard folder structure. All available streams are listed as folder names under the "Data" directory. Each of these streams has subdirectories that represent the corresponding frame, starting with zero. The standard folder structure down to the single frame is illustrated below:

•... \ data \<stream number> \ <frame number> \ - frame-based data record– The frame-based data set now contains all single images, disparity maps, point clouds, and meta information for the given recording location. The sensor number is then marked with the hash «#».

There have 51 segments in this Basel dataset. Different segments have different numbers of streams and frames.

2.1.2 Create the new dataset

In order to decrease the size of the original dataset and use the images that contain building in good view, for each segment, we select the image from the panorama camera and we only use the images in direction 1 and 4 since they are near the camera and usually the camera capture the building from the front side. Direction 1 represents the right side. Direciton 4 represents the left side.

After the filtering, the size of the new dataset, Basel-Panorama-Dataset is around 3000 images in 51 segments. Figure 2.2 shows the images from Basel-Panorama-Dataset. These images are from segment 79368. This segment has 31 frames and the frames 10-17 are shown in the figure.

FIGURE 2.2: Image from Basel-Panorama-Dataset.

These 8 image are from segment 79368, frame 10-17, direction 4.

2.1.3 Rename the image of Basel-Panorama-Dataset

Since dataset images are stored in multi-level folders, It is necessary to extract them in the segment level, which is good for training. After that, we rename the image to record its structure information. For example, image '14053_79367_10_4.jpg', table 2.1 shows the meaning.

TABLE 2.1: Meaning of image file name

Number	14053	79367	10	4
Meaning	segment id	stream id	frame id	direction id

2.2 Zurich rural dataset

This dataset is downloaded from Google Street View, and its location is Zruich rural area. In this dataset, noise images (no building, big area of cars, and green area) have been deleted by the former project and the images containing buildings are left. There are 2141 images in this dataset. Figure 2.3 shows some images from the Zurich rural dataset.

FIGURE 2.3: Image from Zurich rural dataset

2.3 Annotation

VGG Image Annotator(VIA) [16] is a simple and powerful manual annotation software for image, audio and video. We use VIA2.0 tool to annotate these building image images. Figure 2.4 is the interface of VIA tool. By using VIA tool, It is very easy and quick to annotate the label for the image to make the training and validation set.

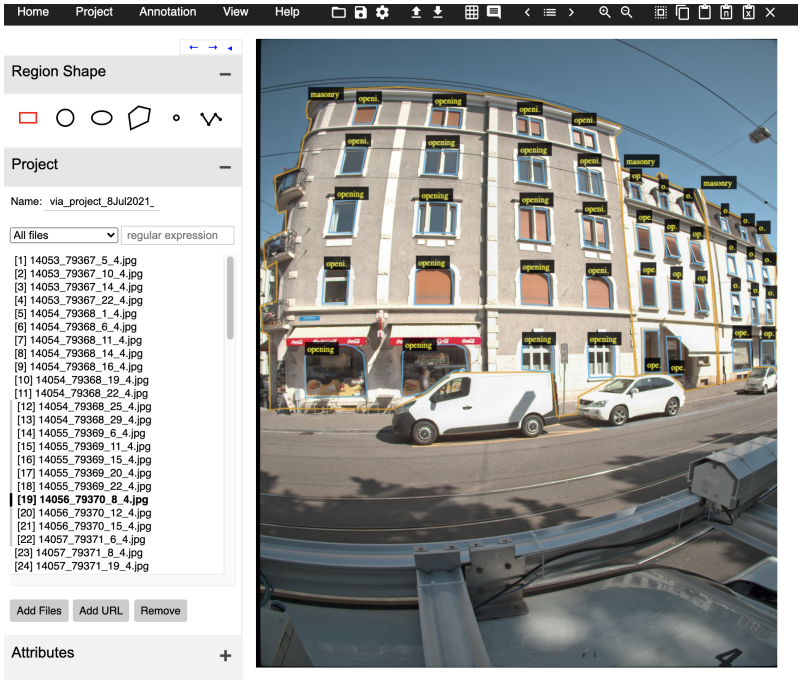


FIGURE 2.4: VIA Tool Interface

In this project, we use a polygon-shape tool to annotate the building types, building openings and building instances. More details will be introduced in the model's dataset section.

Chapter 3

Approach overview

In this project, we need to tackle two tasks, doing the building instance segmentation and predicting the building type. In order to solve this problem. We propose two models to predict them and compare their performances.

Model 1 is a cascaded model. It first uses a fined-tuned mask rcnn model to predict the building and opening(window and door) instance on the input image, then we set the output of mask rcnn model as the input of the building type detection model. This model determines the building type by setting the threshold for the ratio of opening area divided by building facade area. Model 2 is a multi-task model. It just simply use another pre-trained and fin-tuned mask rcnn to detect the building instance and building type at the same time.

Figure 3.1 is the schematic of our approach.

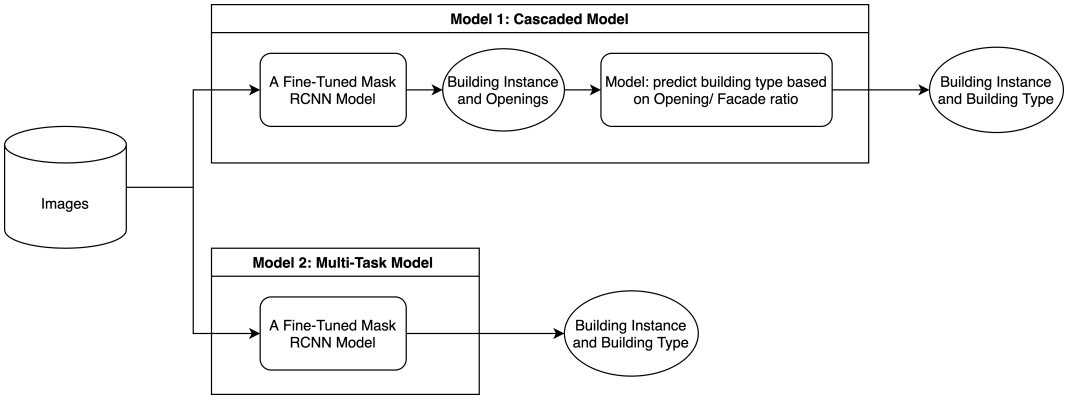


FIGURE 3.1: The schematic of approach: Detect building instance and building type

In the following chapters 4 and 5, Model 1 and model 2 will be explored in detail of the used dataset, implementation, evaluation, and predicted image result, then the comparison and connection between these two models will be discussed in section 5.2.

Chapter 4

Model 1: A Cascaded Model

4.1 Model: Detect building instance and openings

4.1.1 Dataset

Annotation: VIA polygon tool is used to label every opening(window and door) and building in the image. Annotation is a time-consuming work since there are usually having more than 50 openings in one single image.

Train set: 87 images from 24 segments are manually selected as the train set. These images always have a good view of the buildings, which means these buildings in the image are not far from the camera, and most parts of buildings are taken into the image. Examples are shown in figure 4.1.

FIGURE 4.1: Annotated images from Model 1 train set

Validation set: 18 images are selected from 5 segments to make the validation set. These 5 segments are different from segments used in training set. And these images' conditions are the same of images in the training set. Examples are shown in figure 4.2

FIGURE 4.2: Annotated images from Model 1 validation set

Test set: The test set is made of 696 images from the rest 21 segments. They are not labeled. Many of them do not contain the building, and lots of buildings in the image are incomplete and a little far from the camera.

4.1.2 Fine-tuning

Since our training dataset is small and the backbone of mask RCNN is ResNet, a convolutional neural network, usually doing a fine-tuning work based on the pre-trained model can better perform. With fine-tuning, it can help to avoid over-fitting and make the model more robust.

Usually, we fine-tune some higher-level portions of the network. The reason is that the pre-trained model's earlier layers contain more generic and fundamental features (e.g., edge detectors) that could be useful to many tasks, but later or deeper layers become more specific to the details of the classes contained in the original dataset.

In our Fine-Tunned Mask RCNN model, we

1. Use the pre-trained ResNet model training on the CityScapes dataset [17]; The Cityscapes dataset focuses on semantic understanding of urban street scenes. Cityscapes dataset is similar to our dataset since it contains many buildings from street view.

2. Keep the parameter of the backbone ResNet, and fine tune the RPN, classifier, and mask heads of the network.

3. Set the learning rate as 0.00025, which is much smaller than the learning rate 0.02, used in the MaskRCNN paper [7].

4.1.3 Implement

We use Detectron2 platform to train a the pre-trained Mask-RCNN model. Detectron2 is a ground-up rewrite of Detectron that started with maskrcnn-benchmark. The platform is implemented in PyTorch. Detectron2 can run many variants of the models based on the Mask R-CNN structure and it is developed by Facebook AI Group.

There are some key points that need to mention:

- Parameters: Most of the parameters are the same as the original parameters provided by the Detectron2 platform. As for the running epoch, 40 epochs are run on the train set; we set the initial learning rate 0.00025 for the fine-tunning
- Dataset: Since the label of building in the training set is building type, M6, Masonry, RCW. We merge them into a single label, 'building,' and set the output classes name are 'Building' and 'Opening'.

4.1.4 Evaluation on validation set

Mean average precision (mAP) is used to do our model evaluation. Before considering AP value, IoU is an important concept needs to know. Intersection over Union (IoU) measures the overlap between 2 boundaries. It measures the amount of predicted bounding box that overlaps with the ground truth bounding box divided by the total area of both bounding boxes.

$$IoU = \frac{\text{area of overlap}}{\text{area of union}} \tag{4.1}$$

We pre-define an IoU threshold in classifying whether the prediction is a true positive or a false positive. For example, AP_{50} means calculating AP at 0.5 IoU. If $AP_{50} >= 0.5$, we consider the prediction is true positive. Otherwise, it is a false positive. And the notation AP means that AP@[.5:.95] corresponds to the average AP from 0.5 to 0.95 IoU with a step size of 0.05.

Table 4.1 shows the evaluation result for the validation set.

Evaluation	$mAP_{50}(\%)$	$mAP_{75}(\%)$	$mAP(\%)$
Model 1	87.68	52.49	51.11

TABLE 4.1: Model 1 mAP

Categories	Building	Opening
$AP(\%)$	57.311	44.925

TABLE 4.2: Model 1: AP for building instance and opening

FIGURE 4.3: Model 1 validation prediction

Images on the left side are the original one; Images on the right side are the prediction.

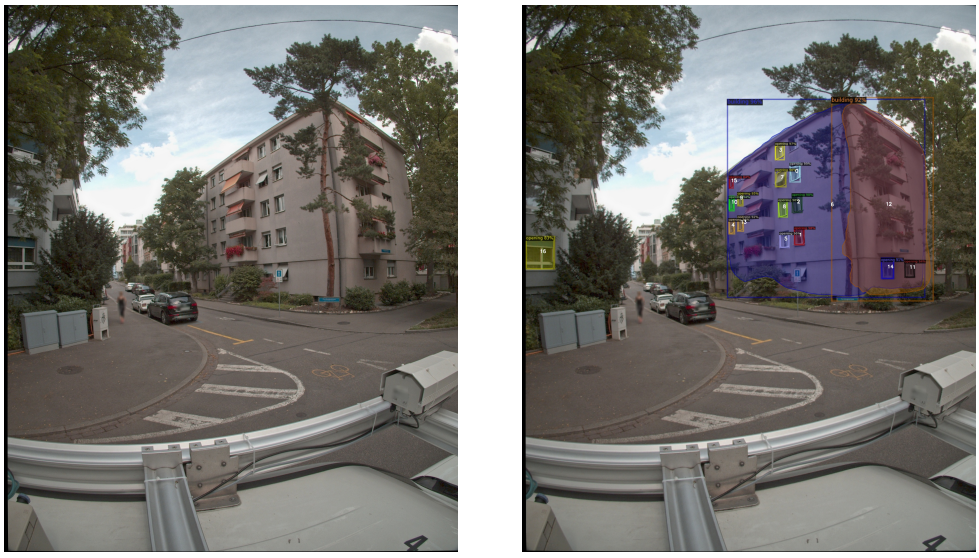
Model 1's mAP_{50} is 87.68%, which is high and good performance. and model 1 has a better performance on predicting the buildings than the opening. Some prediction result for the validation set in figure 4.3. Images on the left side are the original ones. Images on the right side are the prediction. From the prediction, Openings and buildings are perfectly detected, but some openings far from the camera are ignored.

4.1.5 Prediction on test set

Figure 4.4 shows the prediction some results of test image. Compared with the prediction result on the validation set, these problems are often found in the prediction on the test set:

1. **Overlapping**: Masks of two predicted buildings have overlapping areas. This means models predict that some parts of the image belong to two different images. (e.g Figure 4.4.a)
2. **Predict two facades**: This problem is that two facades of one single building are predicted as two distinct buildings. In the training set, only one facade of the building is annotated if it has two facades. (e.g Figure 4.4.b)
3. **Other reasons**: The model can not predict the building correctly. One reason is that the large area of trees, cars, and other stuff covers the front of the building. Another reason is the size training set is small, so the model can not learn enough features from the train set to detect the building. (e.g Figure 4.4.c).

(a) Overlapping



(b) Two Facades

(c) Other Reason

FIGURE 4.4: Model 1 test set prediction

Images on the left side are the original one; Images on the right side are the prediction

4.2 Model: Building type detection

The output of the fine-tuned mask rcnn model above is the building instance and opening. Then we set these output as the input of the model of finding the building type. The main idea of building type detection model is to decide the building type based on value of the opening ratio (Equation 4.2) :

$$\text{opening ratio} = \frac{\text{the area of openings}}{\text{the area of facade}} \tag{4.2}$$

Different types of buildings usually have different opening ratios. Here opening means window and door. In our project, we only consider these three building types: Masonry, M6 [18], Reinforced Concrete Wall (RCW). Their examples are shown in figure 4.5. Masonry building is the typical building in Switzerland, it has a normal ratio of opening. M6 has a little bigger opening ratio than Masonry and usually, it has a relatively large balcony. RCW buildings always have a large size opening. Based on these characteristics, we need to check if there exist the thresholds of opening ratio that can distinguish the three building types in our samples.

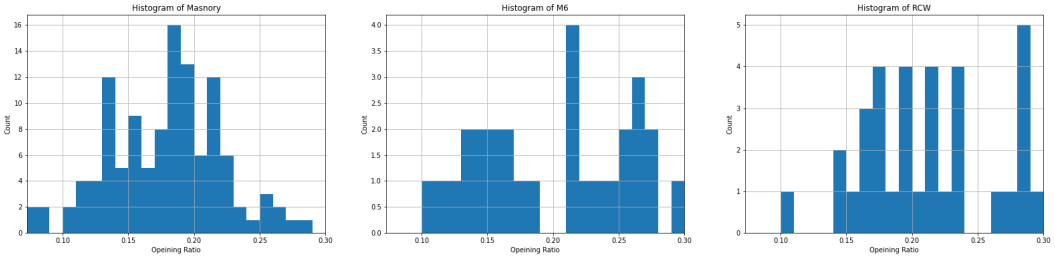
(a) Masonry (b) M6 (c) RCW

FIGURE 4.5: Three building types

We calculated the opening ratio of the building in the 87 images, which are labeled the building type, from the train set. There are 196 buildings (117 Masonry, 33 M6, 44 RCW) in the train set.

Opening Ratio	Masonry	M6	RCW
Mean	0.179	0.229	0.254
Standard Deviation	0.044	0.085	0.098

TABLE 4.3: Opening ratio for three different type in train set



(a) Masonry (b) M6 (c) RCW

FIGURE 4.6: Opening ratio distribution

For three building type in the train set, table 4.3 shows the opening ratio of openings and figure 4.6 shows the distribution of opening ratio.

The mean opening ratio value of Masonry, M6, RCW is 0.179, 0.229, 0.254. These values are indeed in line with the features we discussed above, but when looking at the distribution of the value of the opening ratio, these values are discrete and they have intersections. It is hard to find a threshold to differ the building type. For example, If we set the threshold for Masonry as 0.179-0.229, more than 50% building type will be predicted wrong, so we decide to not use this model to predict the building type.

4.2.1 Other application: detect building floor

Another important topic for the risk assessment is to detect the number of building floors and get the story height. In this project, we predict the number of building floors by design algorithm to connecting the opening's centroid and cluster them into floor groups.

The number of building floor

The basic idea of finding the number of building floors is setting a range of angles to group the opening centroids that meet the conditions. The range of angle is related to the slope sign of the building mask roof line. The reason for using roof slope is that the floor line slope is parallel to the floor. Here are the algorithm steps.

1. Find x coordinate range of the middle 60% points in building bounding box roofline. Set it as X_{range}. With this range, get the corresponding middle 60% roof points, P_{roof} of the building mask.

2. Use linear regression to calculate the slope k of P_{roof} and the sign of k as $sign_k$. Based on the $sign_k$, set the angle range.

3. If $sign_k$<0, set angle range = $(\frac{1}{3} * k - 25, \frac{1}{3} * k + 15)$. If $sign_k$>0, set angle range = $(\frac{1}{3} * k - 15, \frac{1}{3} * k + 25)$. This setting means $\frac{1}{3} * k$ is the baseline. According to the sign of roofline slope, we adjust the angle slope to group the openings.

4. Start from most left side opening's centroid point A. Its x-axis value is X_A; Find the other openings' centroid points whose x-axis value larger than X_A , call it as list L, then order L by x in ascending way; Loop the points D in the L, check if the slope of A and D meet the angle range. If yes, connect and group them, then break the loop, start from D and repeat step 4. If not, continue to loop the points.

5. After finding one group, filter them out, then repeat step 4 to find another group.

For example, in the first image of figure 4.7.a, we want to find the floors of the building on the left. It contains many openings. Following step 4, we start from opening 29, which is the most left opening. Its $L = [17, 25, 27, 21..]$. Then loop the points in L, first 17, pass, and we find 25 and group it. After that, we start from 25 and group the other points.

By using this method, we can clearly and quickly find the number of floors. Figure 4.7 shows the plotting result. The floors in the building are nicely plotted. The evaluation on the train set and val set is on the table 4.4. From the table, we can see the accuracy is very high and this performance is excellent.

Several predictions (e.g figure 4.7.b) are wrong because the complete roofline of building is not taken in the image. Due to this incomplete , the slope of roofline will be calculated incorrectly, leading to an incorrect prediction of the number of floors.

Data Set	Image Number	Image Number (successfully predict #floor)
Validation	18	17
Train	87	81

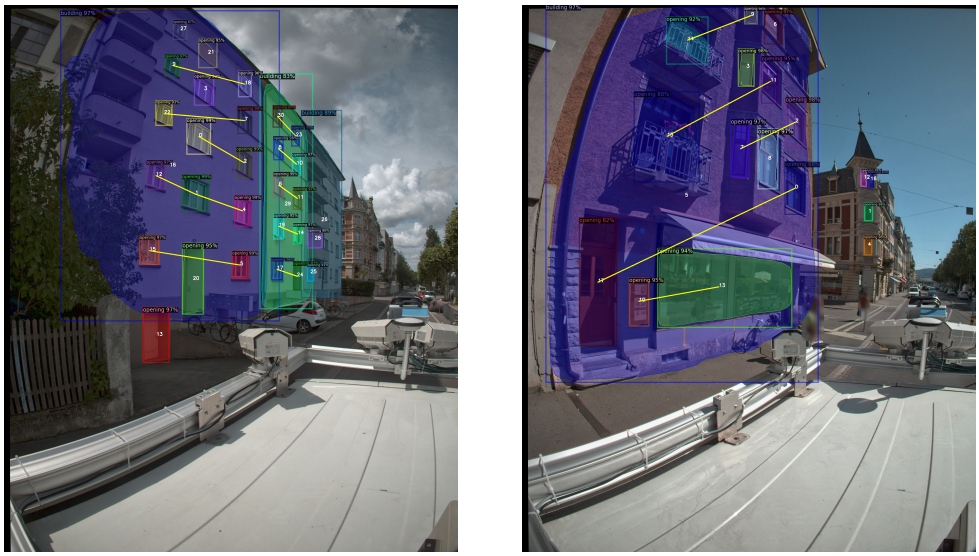
TABLE 4.4: Evaluation of predictiing the number of floors

Storey height

The method of finding the storey height is simple.

- From the above section, we know the openings belong to which floor. One floor's opening group can be noted as List $L = [A, B, C, ...]$. A means the centroid coordinate of the openig A. Then, calculate the centroid of the points in the List L, get the this floor's centroid C

(a) Correct Plotting



(b) Wrong Plotting

FIGURE 4.7: Plot building floors

- After calculateing each floor's centroid, C_i. The y-axis difference between the two adjacent C_i is the Storey Height H.
- H's unit is the pixel. Use the global world coordinate conversion method which is introduced in section 6.1, to get the real storey height.

Chapter 5

Model 2: Multi-task Model

In this chapter, we develop another fine-tuned Mask-RCNN model to predict the building distance and building type. The model structure is similar to the model used in section 4.1. The difference is that we change the potential output class number from 2 to 3. In the model used in section 4.1, the output class is 'Building' or 'Opening', but in this model, the output class becomes the building type 'Masonry', 'M6' [18] or 'Reinforced Concrete Wall (RCW)'.

5.1 Dataset

Annotation: VIA tool is still used to label building types. In model 2, we only concern these three building types, 'Masonry,' 'M6', and 'Reinforced Concrete Wall (RCW).'

Train set: The train set has 279 images. 167 images are from Basel dataset. 112 images are from Zurich dataset. The ratio of 'Masonry', 'M6', and 'RCW' in the training set is roughly equivalent. The most important reason for adding the building from the Zurich dataset is that more RCW building samples are needed since there are only several RCW buildings in the Basel dataset. The other reason is that we can quickly make the size of the training set become double.

Validation set: Val set is made by 52 images. 27 from Basel dataset. 25 images are from Zurich dataset.

5.1.1 Fine-tuning and implement

In the fine-tuning and training part, we set the same parameters as the model used in section 4.1 except the training epoch is 20 and the output class number changing from 2 to 3. . We repeat the other training process and implementation used in the detectron 2 platform to fine-tune a new mask-rcnn model pre-trained on the Cityscapes dataset.

5.1.2 Evaluation and result

For model 2, we still calculate mean average precision (mAP) to do the evaluation. we set IoU threshold is 0.5 and the final mAP evaluation on he validation set is shown at the table 5.1, 5.2

TABLE 5.1: Model 2 mAP

Evaluation	$mAP_{50}(\%)$	$mAP_{75}(\%)$	$mAP(\%)$
Model 2	77.049	53.945	54.224

TABLE 5.2: Model 2 mAP for each building type

Building Type	Masonry	M6	RCW
$mAP(\%)$	52.313	43.810	66.549

From the table 5.1, $mAP_{50}(\%)$ on the validation set is 77.049%, which is decent and from the prediction result(figure 5.1), we can see the building masks and building type are well predicted. And from the table 5.2, It seems that M6 has the worst prediction that may because the sample of m6 in the training set is the smallest. Overall, the prediction of model 2 is both good in mean level and separated class level.

It still has some wrong predictions. The reasons making model prediction incorrect are the same as model 1, such as overlapping, noise, etc. The model's performance can be improved by expanding the training set and adding more M6 samples.

(a) Images from Basel

(b) Images from Zurich

FIGURE 5.1: Model 2 validation set prediction result

5.2 Comparison and connection between model 1 and 2

For the task, building instance segmentation, model 1 has better performance since the mAP values for predicting the building instance for model 1 and model 2 are 57.311% and 54.224%. The images in the train set of model 1 have the opening labels. With opening's features, it may help model 1 to learn the feature of building better.

For the task, building type detection, model 2 can detect the building type well by mask rcnn model, while for model 1, it predicts building type with a bad precision based on the opening ratio.

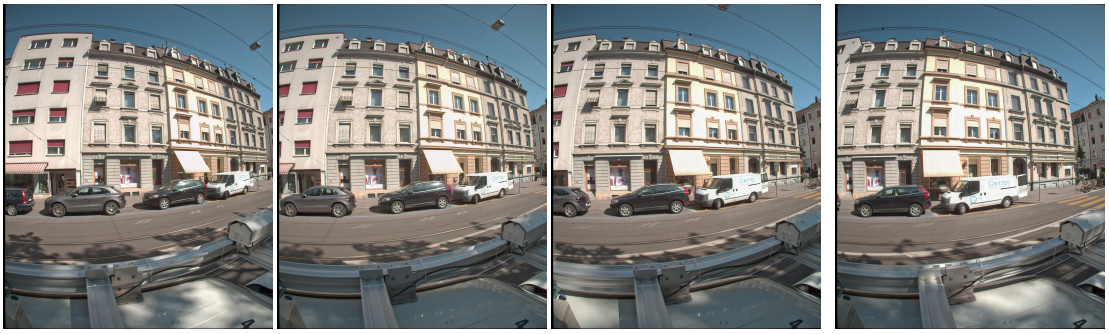
By comparing the performance, we decide to use model 1 to predict the building instance and use model 2 to predict building type. The key step is to ensure the building type matching to the same building instance. In order to finish this linking. The main idea of linking is matching the centroid calculated from the two model's predicted building instances. The detail of algorithm is below:

1. Use image to run model 1. From model 1, we can get the building instances $A = [A_1, A_2, ..A_i]$, then calculate the centroid coordinates for these building instances, note them as $C = [C_1, C_2, ..C_i]$
2. Use the same image to run model 2. From model 2, we can get the building instances $B = [B_1, B_2, ..B_j]$, then calculate the centroid coordinates for these building instances, note them as $D = [D_1, D_2, ..D_j]$
3. For the instance A_i, find the building B_j having minimum $distance(C_i, D_j)$ and $distance(C_i, D_j) < threshold$. If B_j exists, we believe building A_i, B_j are the same building. Here we set the threshold as 400 pixel.

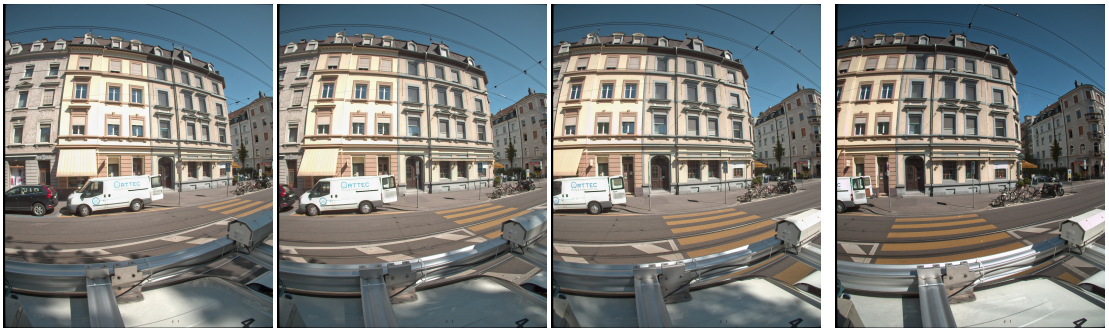
Chapter 6

Building Duplication Detection

Since in the Basel dataset, there are many frames taken in one street, which means that the same building will appear in many images. This kind of situation will lead to duplication problems.



(a) Frame 16 (b) Frame 17 (c) Frame 18 (d) Frame 19



(e) Frame 20 (f) Frame 21 (g) Frame 22 (h) Frame 23

FIGURE 6.1: Images from the stream 79368, segment 14055

To be more specific, the output of model 1 will consist of several cropped buildings that actually are the same building. For example, 8 continuous frame images from stream 79368, segment 14055 Figures contain many same buildings, actually, there are only 4 distinct buildings in the figure 6.1. After using model 1 mask rcnn model to detect the building instance of these 8 images on the images from figure 6.1, we will get lots of duplicated buildings, e.g Figures 6.2 shows one building and its

(a) Frame 16 (b) Frame 17 (c) Frame 18 (d) Frame 19

(e) Frame 20 (f) Frame 21 (g) Frame 22 (h) Frame 23

FIGURE 6.2: One building and its duplication

duplication. In order to count the distinct number of buildings in the street. We need to handle this problem by clustering the duplication.

Our plan to solve this duplication problem by using the following two methods.

1. **Global world coordinate**: This method only considers the coordinate of the building. For each building predicted by model 1, we find its global coordinate. If the distance of two building's global coordinates is smaller than a threshold. We consider them as the same building.

2. **Image-based keypoint matching and clustering**: This method cares about using the keypoint features in the building image. Then using the feature matching between images generate the matrix of the number of matched key points. Then we transform this matrix into a distance matrix. Finally, we use DBSCAN to do the clustering based on this distance matrix.

6.1 Global world coordinate

As introduced in chapter 2, there has a file for each image containing the meta-information of the camera sensor global coordinate. Also, this file only records the

other parameters related to the images and camera, such as camera orientation, distance map. With these parameters, the company that creates this dataset provides a mathematical methodology to convert the image coordinate to global world coordinate. Due to the confidential agreement involved with this company, only the general flow chart 6.3 is introduced here.

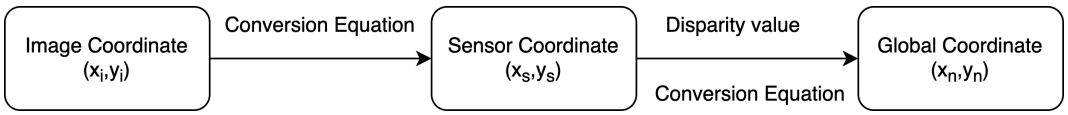


FIGURE 6.3: Global world coordinate calculation

The converted global world coordinate is based on the Swiss LV95 coordinate system. Swiss government introduces the method [19] for converting Swiss LV95 to WGS 84 (World Geodetic System 1984). WGS 84 is a standard for use in cartography, geodesy, and satellite navigation, including GPS and Google Map.

Finding duplication by global coordinate:

1. Assume we have two predicted building instances A, B outputted from model 1. Then find their mask centroid coordinate A_i, B_i.
2. Convert image coordinate A_i, B_i into global world A_n, B_n by using the mathematical methods in figure 6.3
3. Calculate the distance d_n between A_n, B_n and check if the d_n smaller than the threshold T. If yes, these two buildings are recognized as the same building.

While this method is very simple and effective, we meet some problems using the equations to getting the global coordinate and still contact the company to solve it.

6.2 Image-based keypoint matching and clustering

In this section, an image-based building duplication detection method based on the local keypoint features is described. The entire detection procedure can be described as figure 6.4 shown:

From figure 6.4, the duplication detection approach based on Local features (SIFT and DoG-AffNet features) can be summarized as the following steps:

1. Query and test images in the database are put into the system. The local descriptor of query image and database descriptors of database images are extracted by the DoG-AffNet package. It is described in section 6.2.2;
2. Feature matching method is explained in section 6.2.3. It bases on the distance ratio threshold to filter the matching key points.

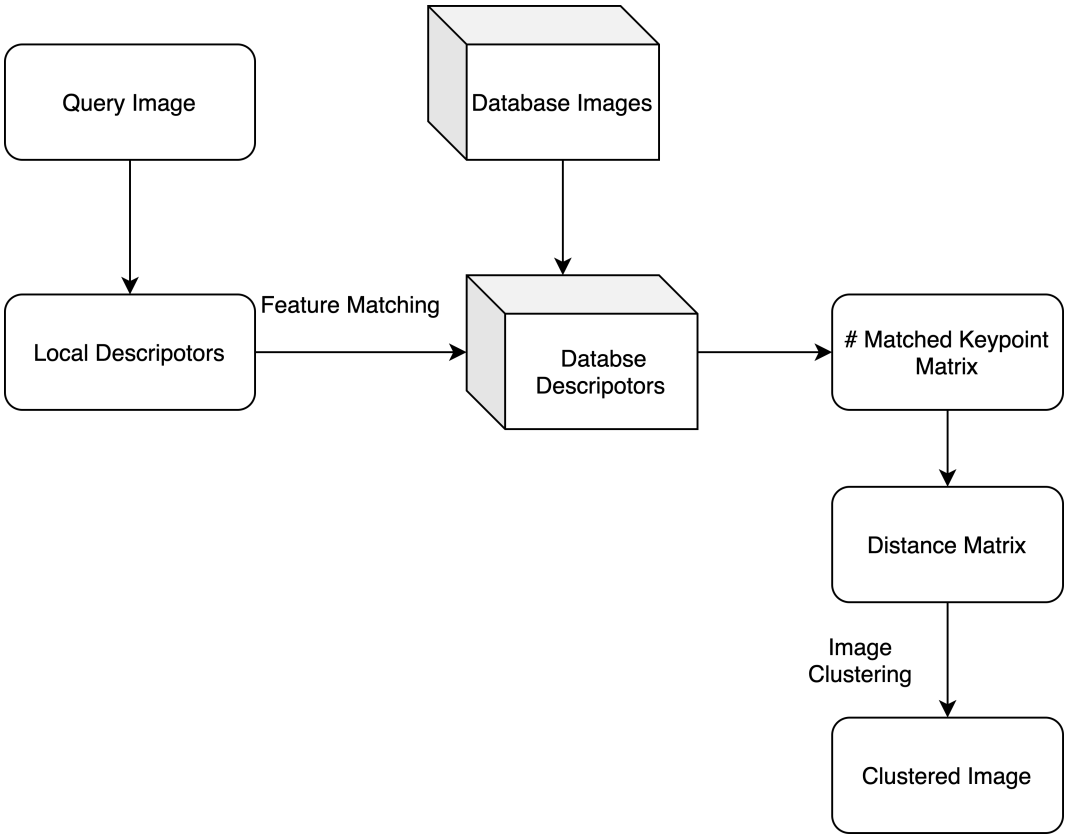


FIGURE 6.4: The flow chart of duplication detection

3. After doing feature matching, several wrongs matched key points still exist. To decrease wrong matched features, RANSAC method is applied to matched key points. It is explained in section 6.2.4

4. After feature matching and RANSAC, we can get a matrix M to show the number of matched key points between the query image and any test image in the database. See section 6.2.5

5. In section 6.2.6, we transform the matrix M to distance matrix D. Then in section 6.2.7, in order to do the image clustering and find the duplication, DBSCAN (Density-based spatial clustering of applications with noise) clustering algorithm is applied in the distance matrix D.

6.2.1 Dataset

First, we select the images from 15 segments. The ids of used segments are shown on the table 6.1. Using model 1 mask rcnn model to predict the building instance on these images and crop the building instance to get the image like figure 6.2. These cropped images from 15 segments make the test dataset for duplication detection.

6.2.2 Local descriptors

In our project, we use two local descriptors, SIFT and DoG-Affnet, to find the features of the image. There is basic knowledge of these two local descriptors below.

SIFT

Scale Invariant Feature Transform(SIFT) [20] is an approach for detecting and extracting local feature descriptors which are invariant to image illumination, scaling and rotation. In 2004, D.Lowe, came up with this new algorithm, Scale Invariant Feature Transform (SIFT).

Detection stage of SIFT feature is divided into 4 steps [21]:

1. Scale-space Extrema Detection

Image I(x, y) is convolved with Gaussian filters at different scales as equation 6.1:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{6.1}$$

In the equation 6.1, L(x, y, σ) is the convolution of image I(x, y) with the Gaussian filter G(x, y, σ) at scale σ. Differences between two Gaussian images at scale kσ and σ are taken as equation 6.2 shown:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{6.2}$$

the difference at these two scales are called a DoG(Differences of Gaussians), image of which is presented in figure 6.5 :

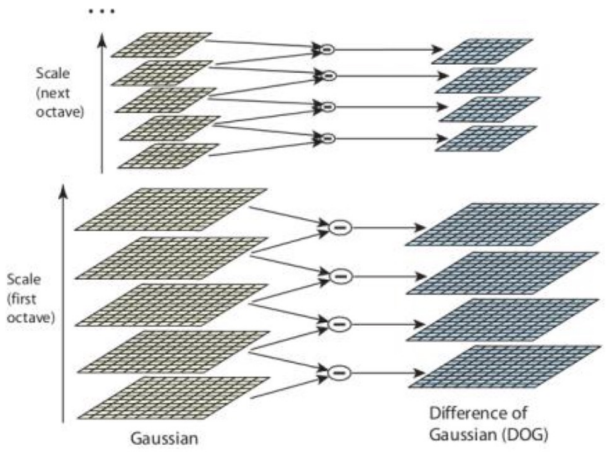


FIGURE 6.5: The creation of a DoG image

2. Keypoint Localization

After the first stage, keypoints, also can be called Interest points are identified as local maxima or minima of the DoG images across scales. Each pixel in the DoG

images is compared to its 8 neighbours at the same scale, which is shown in figure 6.6. We also need to accurate the keypoints' localizations by discarding points by a predetermined value.

$$D(\hat{x}) = D + \frac{1}{2} * \frac{\partial D^T}{\partial a} * \hat{x} \tag{6.3}$$

where $\hat{x}$ is calculated by setting the derivative D(x, y, σ) to zero

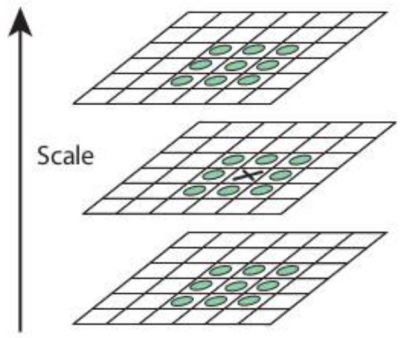


FIGURE 6.6: Local extrema detection

3. Orientation Assignment

In order to achieve invariance to orientation, the gradient magnitude m(x, y) and orientation θ(x, y) are precomputed using as equation 6.4, 6.5

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \tag{6.4}$$

$$\theta(x, y) = \arctan(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y}) \tag{6.5}$$

4. Keypoint Descriptor

If a keypoint orientation is selected, the feature descriptor is computed as a set of orientation histograms on 4 $\times$ 4 pixel neighbourhoods, it is shown in figure 6.7:

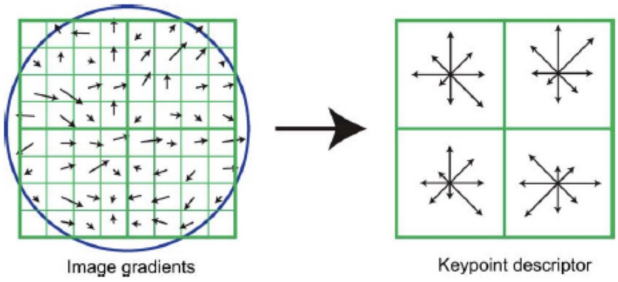


FIGURE 6.7: SIFT feature descriptor

Histogram contains 8 bins, so a SIFT feature vector is with $4 \times 4 \times 8 = 128$ elements,and it is normalized to unit length. It's the representative of SIFT feature descriptor.

DoG-AffNet

AffNet is a novel method for learning local affine-covariant regions. The affine shape estimator – AffNet – trained with the hard negative-constant loss outperforms the state-of-the-art in bag-of-words image retrieval and wide baseline stereo. [22]

Affine-convariance [23] is a desirable property of local features since it allows robust matching of images separated by a wide baseline, unlike scale-covariant features like ORB [24] or difference of Gaussian (DoG) [21] that rely on tests carried out on circular neighborhoods.

Affnet architecture is adopted from HardNet [25], see figure 6.8 , with the number of channels in all layers reduced 2x and the last 128D output replaced by a 3D output predicting ellipse shape.

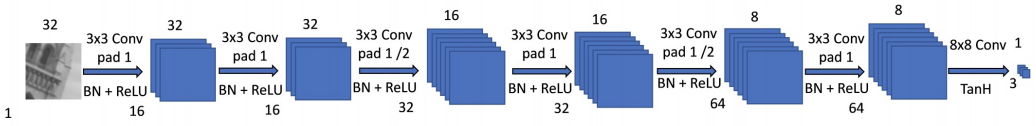


FIGURE 6.8: Affnet architecture

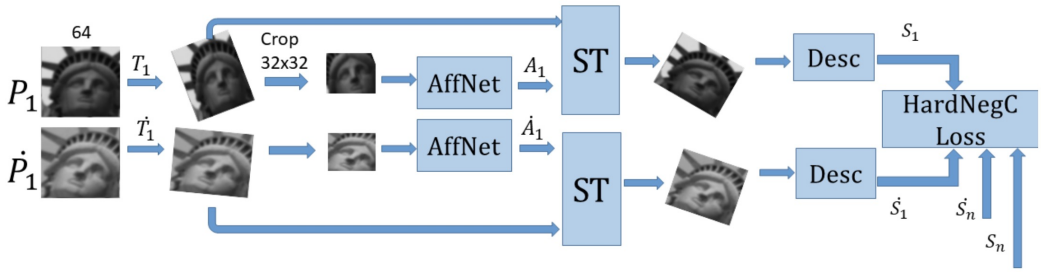


FIGURE 6.9: Affnet training

Affnet training steps: use corresponding patches from UBC Phototour dataset, undergo random affine transformation $T_i, \dot{T}_i$, are cropped and fed into AffNet, which outputs affine transformation $A_i, \dot{A}_i$ to an unknown canonical shape. ST – the spatial transformer warps the patch into an estimated canonical shape. The patch is described by a differentiable CNN descriptor. $n \times n$ descriptor distance matrix is calculated and used to form triplets, according to the HardNegC loss [22].

Implement: Using Open CV can easily extract the SIFT keypoints. Figure 6.10 show the feature keypoints by using SIFT. The example of AffNet keypoints is in the figure 6.12.a.

(a) Before SIFT feature extraction (b) After SIFT feature extraction

FIGURE 6.10: SIFT feature extraction by using OpenCV

6.2.3 Feature matching

Based on the introduction about descriptors, we use these two local descriptor methods to extract the key points, then do the feature matching.

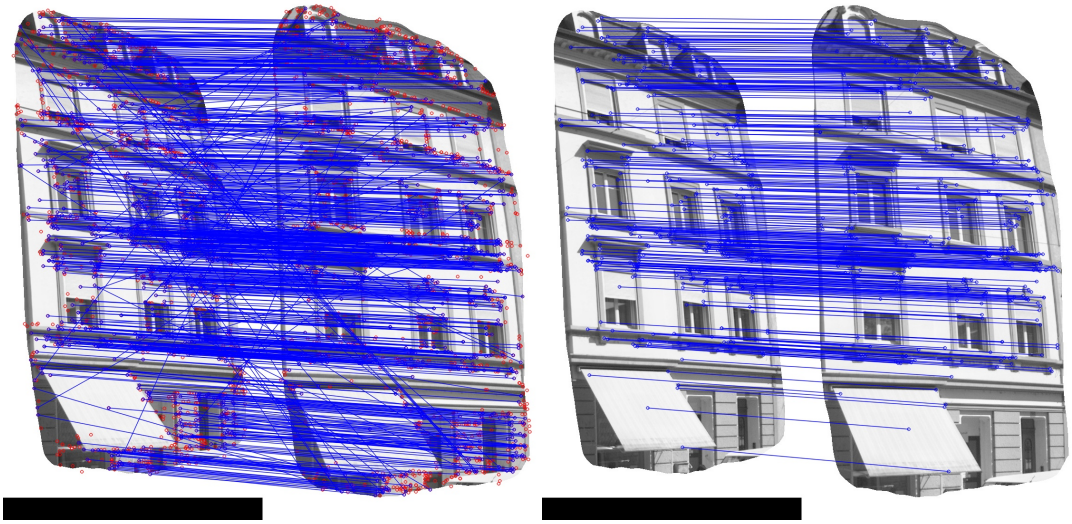
In order to find matched pairs of features points located in the query image and the test image. *Low* developed nearest neighbor method [21], which is effective and useful on the feature matching problem. The best candidate match for each keypoint is found by identifying its nearest neighbor in the database of keypoints from query images.

Here, we select one SIFT feature point from the query image, name this point as F_{query} and name its feature descriptor as d_{query}. Next, find d_{query}'s nearest and second nearest feature descriptor from features in the test image based on the Euclidean distance in feature space (128 dimensions). These two features are called F_{first} and F_{second}, and their feature descriptors are called d_{first} and d_{second}. And we calculate the ratio between the two distances which is shown in equation :

$$ratio = \frac{\text{Euclidean Distance}(d_{query}, d_{first})}{\text{Euclidean Distance}(d_{query}, d_{second})} \tag{6.6}$$

According to Lowe's paper, when the ratio of the two distances is less than a threshold, it means two feature points are matched. Also, it means that feature F_{query} and F_{first} are matched. Based on the paper, when the ratio is 0.8, feature matching gets the best performance [21]. In our experiment, we initially set the ratio as 0.8.

Based on the feature matching method described above. We used SIFT, and DoG-Affnet descriptors to extract the features. In this method, Figure 6.2.a is considered



(a) Feature Matching by SIFT (b) Feature Matching by SIFT after RANSAC

FIGURE 6.11: SIFT feature matching between query image and test image

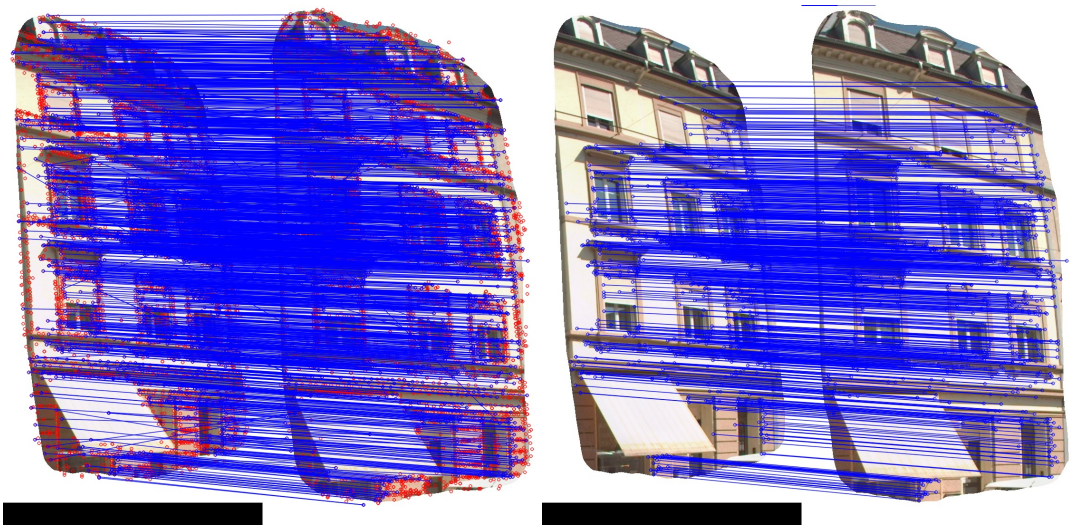
as the query image. Figure 6.2.b is considered as a test image. Matched pairs of feature points located in the query image and test image shown in the figure 6.11.a, 6.12.a. Those matched feature points are linked with blue lines. For the SIFT method, 795 matched key points are found by using ratio test, while using DoGnet found more points, 1463 keypoints are matched. It shows that DoG-AffNet has a better performance. In the later section, we only consider DoG-AffNet since it performs better.

6.2.4 Matching correction by RANSAC

After doing feature matching in our example, there are still having many incorrect matching feature pairs. In order to delete these incorrectly matched feature points, we use random sample consensus (RANSAC) method.

Random sampling consensus (RANSAC) is an iterative method of estimating mathematical model parameters by using observed data points. The data points include inlier and outlier. Outlier has no effect on the estimation of the model, so this method can also be called outlier detection method. This is a non-deterministic algorithm because it obtains a reasonable result under a certain probability. When the number of iterations increases, the probability will increase. This RANSAC algorithm [26] was first proposed by Fischler and Bolles in 1981.

Here we used the same query and test image as figure shown in the figure 6.11.b, 6.12.b. After using RANSAC method, we can find the many mismatched keypoints are deleted and the left key points clearly show a similar region of query and test image. For SIFT, the number of matched keypoint decreases from 795 to 396. For DoG-AffNet, it decreases from 1463 to 742.



(a) AffNet feature aatching without RANSAC (b) AffNet feature aatching with RANSAC

FIGURE 6.12: AffNet feature matching between query image and test image

6.2.5 Matching matrix

After doing feature matching, we can use the number of matched keypoint to represent the similarity between the query image and test image. In order to figure out the similarities between the query and the other images in the database, we use a matching matrix, M to record them.

The flow chart 6.13 presents the procedure to generate the matching matrix M, The process can be concluded like this. For each image i, we use the AffNet feature matching to generate the Array i(Dimension 1*N). After looping each image i, we get the N array, then merge these N arrays to get the final matching matrix M (Dimension N*N).

6.2.6 Distance matrix

A typical method to do image clustering is based on the distance matrix. distance matrix represents the distance between any two features extracted from the two images. while the matching matrix M is a kind of similarity matrix, we need to transform it to distance matrix D. The method we use is below.

1. Note each element of the matching matrix M as m_{ij}. Note each element of the distance matrix D as d_{ij}. Doing the option below.

$$m_{ij} - - > \frac{1}{a_{ij}} = d_{ij}$$

2. If $m_{ij} = 0, d_{ij} = inf$, reset $d_{ij} = 1$. This means that the maximum distance between two images is 1.

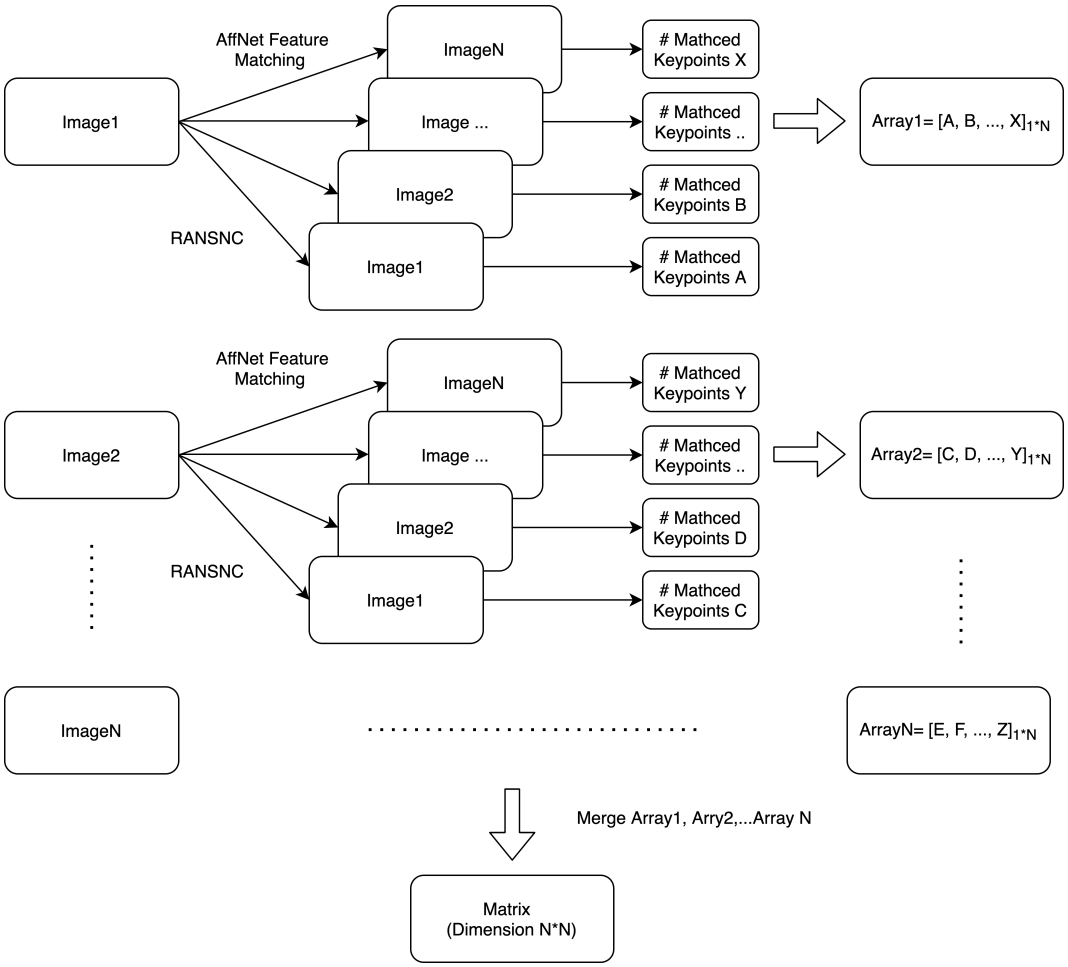


FIGURE 6.13: Generate Matching Matrix M

3. Reset the diagonal element of D to 0, $d_{ii} = 0$. This means the distance between the same image is 0.

6.2.7 Image clustering - DBSCAN

DBSCAN algorithm

The last step of finding the building image duplication is to clustering the images based on the distance matrix D. In this project, we can not know the number of clusters in advance. One popular clustering technique that does not require the specification of the number of clusters is DBSCAN.

DBSCAN stands for Density-based spatial clustering of applications with noise. This algorithm can identify clusters in large spatial data sets by looking at the local density of based database elements, based on the distance between observed data.

In the DBSCAN algorithm, there are two key parameters: eps and min_samples. eps represents the maximum distance between two samples for one to be considered as in the neighborhood of the other. min_samples defines the number of samples (or total weight) in a neighborhood for a point to be considered as a core point. This

includes the point itself. It assumes that a cluster is a dense region with data points that is greater than min samples within the range of eps of the core point and each cluster is separated from another by lower density.

Compared with other clustering algorithms, DBSCAN method has the following advantages:

1. The number of clusters does not need to be defined in advance, unlike K-means, spectral clustering, and hierarchical clustering method need to pre-define it.
2. It is not sensitive to the outliers, and outliers can be ignored while doing clustering.
3. It is possible to cluster dense data sets of any shape. In contrast, clustering algorithms such as K-Means are generally only suitable for convex data sets.

Implement

- Set min_sample as 4. After having a deep digging into the cropped building images, we found one single image usually has more than 4 duplications (include itself). we ignore the building if its duplication number is smaller than 4.
- Set eps in range (0.2,0.6,0.02). Loop the DANSCAN method in this range to find the eps that make the predicted clusters having maximum cluster numbers. At the same time, we keep the size of the noise group (in the code, noted as group '-1')as small as possible.

With the algorithms and parameters setting above, we test the performance on several segments. These test images are the cropped building image and coming from the same segment. They are the result of the model 1. The bounding box of building instance, instance mask, and building type is predicted from the multi-mask model, then we crop the bounding box to get the cropped building image and set the background outside the building mask as a white pixel.

Evaluation and result

Figure 6.14 shows a nice clustering result for the cropped building instance in segment 16878. 24 cropped instances are grouped into 4 clusters by DBSCAN, and none of the building instances is incorrectly clustered.

(a) Cluster 1

(b) Cluster 2

(c) Cluster 3

(d) Cluster 4

FIGURE 6.14: Clustered buildings from segment 16878

TABLE 6.1: DBSCAN Evaluation: Purity

Segments	Predicted #Cluster	True #Cluster	Purity Ratio	Purity Value
16878	4	4	24/24	1
16879	5	5	28/29	0.966
16948	5	9	58/122	0.475
16949	7	10	46/77	0.597
16875	3	7	35/63	0.547
16880	1	1	6/7	0.857
16881	1	1	7/8	0.875
16882	3	4	31/31	1
16883	2	2	14/14	1
16884	10	8	72/98	0.735
16885	2	3	82/98	0.837
16888	3	3	29/29	1
16889	15	20	145/190	0.763
16890	13	15	116/162	0.716
19416	3	3	49/49	1
All	77	90	742/1002	0.741

Table 6.1 shows the clustering result in several segments.

We choose to use the purity index to evaluate the performance of the DBSCAN clustering model. Purity is a measure of the extent to which clusters contain a single class. Its calculation can be thought of as follows: For each cluster, we count the number of data points from the most common class in said cluster. Now take the sum over all clusters and divide by the total number of data points. Formally, purity can be defined as equation 6.7:

$$\text{Purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{k} max_j |\omega_k \cap C_j| \tag{6.7}$$

where, N represents the total number of samples, $\Omega = \{\omega_1, \omega_2, \omega_3, ...\omega_k\}$ represents the predicted clusters, and $\mathbb{C} = \{C_1, C_2, C_3, ...C_j\}$ represents the true clusters.

The range of purity value is $[0, 1]$. The greater purity means the better clustering performance. DBSCAN find all the true clusters and perform very well since for the most test segments, the purity is higher than 0.7. And overall purity is 0.741.

However, there are still some wrong clusters. It should be noticed that fewer clusters are grouped for the segments 16948, 16949, 14875, 16889, and more clusters are predicted for segment 16884. After looking at all the clustered images, we conclude three reasons for the wrong clustering.

- **Several facades for same building** : DBSCAN may cluster the front, left, and right facade of the same building as the distinct building. This is reasonable since the different facade has different features. e.g Figure 6.15 In segment 16884, 2 facades of the same building are predicted as two clusters.
- **Similar building structure**: Buildings that contain big areas of windows and have a balcony sometimes are cluster into the same group. Since these buildings have these similar structures, they may have a large number of matched keypoints when doing the affnet feature matching. In other words, these images distance are always small and similar in the distance matrix D, e.g Figure 6.16

(a) 16884, front side facade

(b) 16884, right side facade

FIGURE 6.15: Two facades predicted as two buildings

FIGURE 6.16: Similar structure predicted as same building

6.2.8 Building type prediction after clustering

After clustering the building duplication by DBSCAN method, the distinct buildings from each segment are found. The next task is to find the building type (Masonry, M6, RCW) of the distinct buildings.

We used model 2 mentioned in chapter 5 to predict the building type. Then follow the method that matching the centroid coordinate of buildings to link the building type to the building instance, which have discussed in section 5.2. Finally, we set the most frequent type in each duplicated cluster as the final prediction of building type for this cluster. Here one cluster represents one distinct building.

TABLE 6.2: Building type prediction after clustering

Segments	Building Number	Masonry	M6	RCW
16878	4	1	2	1
16879	5	0	5	0
16948	5	3	1	1
16949	7	4	3	0
16875	3	2	1	0
16880	1	0	1	0
16881	1	0	1	0
16882	3	1	2	0
16883	2	0	2	0
16884	10	0	9	1
16885	2	0	1	1
16888	3	0	3	0
16889	15	1	12	2
16890	13	6	3	4
19416	3	0	0	3
All	77	18	46	13

The building type prediction result on the test segments after clustering is shown in table 6.2. In the 15 testing segments from the Basel area, we predict there are 77 buildings (18 Masonry, 46 M6, 13 RCW) located in this area. And the building type ratios for Masonry, M6, and RCW are 23.4%, 59.7%, 16.9%.

Chapter 7

Conclusion and future work

In this report, two models are introduced to predict the building instance and building type. Model 1, a cascaded model, gets a mAP 57.311% on predicting the building instance, and it fails to find the building type by using the opening ratio threshold. The performance of model 2, a multi-task model, on predicting the building instance, is a little worse than that of model 1. Its mAP is 54.224% . Model 2 can predict the building type at the same time when predicting the building instance.

After comparing the performance of these two models, we decide to use model 1 to predict the building instance and use model 2 to get the related building type. The method to connect these two models is by checking if the centroid coordinates of two building instances predicted by two models are close.

Then, we used the DBSCAN method to do the building image clustering based on the feature distance matrix to solve the building duplication problem. The evaluation of clustering is getting the purity, 0.741 on the test set. Finally, we predict there are 77 buildings on the 15 testing segments. Their building types are 18 Masonry, 46 M6, and 13 RCW.

As for future work, we need to give priority to handling the problems of converting the image coordinate to the global world coordinate. This conversion is crucial for showing the building type and location of the distinct buildings on the map.

Next, we will try to expand the train and validation set of model 1 and model 2. This action may improve the model performance and solve part of the wrong prediction since the model can learn more features from the larger size of samples. We also try to use the 3D point cloud, which is already provided, to do the instance segmentation and object classification.

Bibliography

[1] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *European conference on computer vision*, pages 297–312. Springer, 2014.

[2] Bernardino Romera-Paredes and Philip Hilaire Sean Torr. Recurrent instance segmentation. In *European conference on computer vision*, pages 312–329. Springer, 2016.

[3] Anurag Arnab and Philip HS Torr. Bottom-up instance segmentation using deep higher-order crfs. *arXiv preprint arXiv:1609.02583*, 2016.

[4] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[5] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[8] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019.

[9] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12193–12202, 2020.

[10] Jian Kang, Marco Körner, Yuanyuan Wang, Hannes Taubenböck, and Xiao Xiang Zhu. Building instance classification using street view images. *ISPRS journal of photogrammetry and remote sensing*, 145:44–59, 2018.

[11] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[12] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[13] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[15] Yang Yu, Kailiang Zhang, Li Yang, and Dongxing Zhang. Fruit detection for strawberry harvesting robot in non-structural environment based on mask-rcnn. *Computers and Electronics in Agriculture*, 163:104846, 2019.

[16] A. Dutta, A. Gupta, and A. Zissermann. VGG image annotator (VIA), 2016.

[17] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.

[18] Sergio Lagomarsino and Sonia Giovinazzi. Macroseismic and mechanical models for the vulnerability and damage assessment of current buildings. *Bulletin of Earthquake Engineering*, 4(4):415–443, 2006.

[19] Swiss Government. Approximate formulas for the transformation between swiss projection coordinates andwgs84. 2016.

[20] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, page 1150, USA, 1999. IEEE Computer Society.

[21] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–, 11 2004.

[22] Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Learning discriminative affine regions via discriminability. *CoRR*, abs/1711.06704, 2017.

[23] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004.

[24] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[25] Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. *arXiv preprint arXiv:1705.10872*, 2017.

[26] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.