Leibniz Universität Hannover

Masterarbeit Jianqi Sun

Water Level Estimation for Vehicles Submerged in Flood Water

Betreuer: Dr.-Ing. Yu Feng

Prüfer: Prof. Dr.-Ing. habil. Monika Sester

Prof. Dr. techn. Franz Rottensteiner

ikg

Water level estimation for vehicles submerged in flood water

Master thesis at the ikg

Introduction and goal of the thesis

Flooding is a huge challenge for many cities today. The extraction of water level observations is necessary for a more efficient emergency response and disaster analysis. These observations can be used to enhance the situational awareness of city managers and citizens. Objects such as cars submerged in flood water can provide a rough indication on the flood severity.

The overall goal of this master's thesis is to provide water level estimates for images containing cars submerged in flood water. The poses of the cars need to be estimated from images, which can be taken from social media or new articles. Water level estimation at the decimeter level needs to be achieved with the necessary evaluation of the pipeline.

Tasks and time schedule

1. Literature research
2. Implementation of car keypoint detection and car pose estimation
3. Development of water level estimation algorithm
4. Evaluation of components of the pipeline
5. Documentation and discussion

Tools

- ApolloCar3D dataset for autonomous driving research
- Images dataset for flood level estimation from ETHZ

Requirements

- Programming experience with Python, Pytorch, and OpenCV
- Basic understanding about deep learning
- Knowledge in image processing and experience with 3D data

Contact person

Yu Feng (email yu.feng@ikg.uni-hannover.de , Tel. 511-762-19437)

Institute of Cartography and Geoinformatics, Appelstraße 9a, 30167 Hannover

Statutory Declaration

I, Jianqi Sun, declare that this master's thesis, and the work indicated herein have been composed by myself, and any sources have not been used other than those specified. All the consulted published or unpublished work of others have been clearly cited. I additionally declare that the work and master's thesis have not been submitted for any other previous degree examinations.

Jianqi Sun — Hannover, February 17, 2022

Eidesstattliche Erklärung

Ich, Jianqi Sun, erkläre hiermit, die Arbeit selbstständig verfasst zu haben und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, habe ich als solche kenntlich gemacht. Die Arbeit wurde in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt.

Jianqi Sun — Hannover, February 17, 2022

Acknowledgements

The last thesis of the master's degree stage is finished, which means that the master's stage at Uni Hannover is coming to an end. First of all, I would like to thank all the people who have helped me with my thesis, especially Dr. Feng for his careful guidance during my thesis process. I would also like to thank my professors for assigning me a this thesis topic and giving guidance.

I would also like to thank my family and friends who have encouraged me in my life. Especially my girlfriend Yimin, who was always by my side and gave me great spiritual support.

During my four years in Germany, I learned not only professional knowledge, but also academic seriousness and conscientiousness of German people. These valuable qualities will accompany me in my future life and remind me to be a rigorous person at all times.

Finally, I would like to thank all the people who have helped me. Thank you.

Abstract

Due to the frequent occurrence of flood disasters in recent years, dynamic determination of flood extent or flood depths can effectively reduce the impact of flood disasters on lives. Current water level estimation is usually based on contact or non-contact devices, such as staff gauges and ultrasound water level gauges. However, these devices are not highly flexible, such as they need to be placed in fixed locations or require manual interpretation. Therefore, finding a suitable method for water level estimation is an important topic nowadays.

With the rise of computer vision, there are many studies on image-based water level estimation. In this thesis, a combination method of deep learning and traditional computer vision for water level estimation is used. In this method, the flooded cars are used as reference targets. The water level is estimated by the submerged car. First, the Mask R-CNN model is pre-trained on COCO dataset and is used to extract the cars exposed to the flood. Moreover, the corresponding masks for the cars are generated. Then the Keypoint R-CNN model is used to detect the keypoints of the cars exposed to the water part after re-training with the ApolloCar 3D dataset. There are 79 3D models provided in this dataset. The keypoints of the 3D models are labeled manually in this thesis. When the 2D keypoints of the car on the image and the corresponding 3D keypoints in the world coordinate system are abtained, it becomes a PnP problem. The 6 DoF pose information of the 3D model relative to the camera coordinate system can be obtained with the PnP solver. Finally, the iterative method is used to simulate the flooded scene. The 3D model is gradually truncated horizontally from bottom to top. Then the truncated model is projected onto the 2D image. As more and more parts of the 3D model are truncated, the mask projected onto the 2D image becomes closer to the mask generated by Mask R-CNN. Next, the IoU of the two masks is calculated. As long as the lower edge of the 3D model does not reach the water surface of the flood, the IoU becomes larger as more parts of the 3D model are truncated. The IoU stops growing when it reaches the water surface of the flood. At this point, the truncated height in the 3D model is the depth of the flood. By this method, the accurate water level information can be obtained.

Contents

List of Acronyms

CNN	Convolutional Neural Network
FN	Feedforward Neural Network
R-CNN	Region-based Convolutional Neural Network
IoU	Intersection of Union
PnP	Perspective-n-Point
DLT	Direct Linear Transformation
EPnP	Efficient Perspective-n-Point
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

List of Figures

List of Tables

1 Introduction

In recent years, floods have occurred frequently, such as the July 2021 large flood in Henan Province, China, which significantly impacted the economy, infrastructure, and human activities. A recent study on global flood hazard analysis was published online as a cover article in the August 2021 issue of Nature [6]. The results show that the proportion of people at risk of flooding increases globally every year. Accurate estimation of flood growth extents and calculation of flood depths are effective ways to prevent flood disasters.

According to the current research on flood estimation, there are mainly three methods for water level estimation [7].

- Measuring water levels with contact and non-contact type gauges.
- Flood inundation mapping using digital elevation models.
- Image processing and computer vision-based methods.

(a) (b) (c)

Figure 1.1: Methods of water level estimation (example images under CC BY-NC-SA 2.0)

Examples of the above three types of methods are shown in Figure 1.1. Here, (a) measures the water depth by staff gauges, which is a contact gauge to estimate the water depth. In addition to this, there are non-contact methods of measuring water depth by ultrasonic gauges and other methods. The advantage of this method is that the water depth can be measured accurately, but it requires many human resources for on-site measurement. Besides, some contact gauges need to be put in a fixed location and cannot measure the

water depth in the unknown area. In (b) is satellite maps to estimate the extent of flooding. Floods are predicted by changes in extent at different moments in time. This method does not provide accurate and precise water depths, and the trend can only be estimated from the flood extent. And (c) is the method used in this thesis. It is an image-based water level estimation method. The advantage of this method is that the resources are widely available and easy to obtain—for example, surveillance cameras along with the street, Internet, and field photos.

1.1 Motivation

The motivation of the proposed image-based water level estimation method is to estimate the flood water depth by scene of the submerged cars through a wide range of resources.

In this thesis, the flood depth is estimated by combining traditional computer vision methods and deep learning. The main idea is to take the submerged vehicles as a reference target and estimate the flood depth indirectly by calculating the height of the submerged car.

The main steps is as follows. Firstly, the exposed part of the car is detected by Mask R-CNN. Keypoint R-CNN is used to detect the keypoints of this part of the car body. The 6 DoF pose information is estimated using PnP by aligning these 2D keypoints with the keypoints on the corresponding 3D model. Finally, the flood inundation 3D model is simulated to fit with the Mask R-CNN detected car to predict the water level.

1.2 Contribution

The work of this thesis differs from previous studies using submerged vehicles to estimate water level in the following aspects:

- The method is proposed to estimate the flood water level by the submerged cars—this is a multi-stage method for estimating water level by combining deep learning and traditional computer vision methods.
- The proposed method is more generalized compared to other current research methods. For any image of a flood scene with vehicles, information of the flood water depth can be obtained without obtaining the camera intrinsic parameters, only the fake camera intrinsic parameters are required.
- The ApolloCar 3D dataset is processed to generate a dataset of simulated floods, and the neural network model is retrained. With these datasets, the detection rate of the model for submerged cars is improved.

- Images with vehicles in the ETHZ dataset are selected and re-labeled with water level information and location information on the images.
- The keypoints of the three different types of 3D models are annotated manually. This makes the 3D models can be accurately reprojected to 2D images.

1.3 Organization of the Thesis

This thesis consists of the following chapters:

In Chapter 2, the theories of deep learning and traditional computer vision, main datasets and evaluation matrics involved in this thesis will be introduced.

Chapter 3 collates current research on flood level estimation using different reference targets. Different methods are briefly described, and their advantages and disadvantages are analyzed.

In Chapter 4, the details of the approach proposed in this thesis, the treatment of different datasets and the attempts made on various methods during the study are described.

Chapter 5 presents the selection of model parameters in the process of method proposal and the comparison experiments of different methods.

In Chapter 6, the results generated in the experiments are summarized and discussed.

Finally, Chapter 7 summarizes this thesis, discusses the interesting findings, and points out some potential research directions for future work.

2 Basics

Some basic concepts and fundamental methods will be explained in this chapter to understand this thesis better. The first section will introduce the knowledge related to deep learning covered in this thesis. In the second section, some basic methods for pose estimation will be presented, divided into two parts: deep learning-based methods and traditional computer vision methods. The last section will explain the evaluation metrics used in this thesis.

2.1 Deep Learning

Machine learning incorporates techniques from several fields [8], such as probability theory, convex analysis, statistics, brain science, and approximation theory. It is used with computers to simulate or implement human learning behavior through bionics. By acquiring new knowledge or skills and continuously reorganizing the existing knowledge structure to improve its performance.

Deep learning [8] is a branch of machine learning. By using deeper neural networks, more complex learning tasks are achieved. With the rise of artificial intelligence and chip computing power, deep learning is widely used in academic research and industrial production. It is designed for numerous fields, such as computer vision and natural language processing.

The two models, Mask R-CNN and Keypoint R-CNN are applied in this thesis. They are the well-known models of deep learning in the field of computer vision. In the next subsections the basics, the framework of these two models and the applied datasets are described in detail.

2.1.1 Convolutional Neural Network

Convolutional Neural Network (CNN) is a class of Feedforward Neural Networks (FNs) with convolutional computation and with deep structure. It is one of the representative deep learning algorithms. Convolutional neural networks are capable of representation learning and shift-invariant classification of the input information according to their hierarchical structure, so they are also called "shift-invariant artificial neural networks."

The Convolutional Neural Network Architecture consists of three main layers: convolutional layer, pooling layer, and fully connected layer. The convolutional layer helps to abstract the input image into a feature map through filters and kernels. The pooling layer helps to subsample the feature map by summarizing the presence of features in each patch of the feature map. The fully connected layer connects each neuron of one layer to each neuron of another layer.

The layers of a CNN are combined. enables the designed neural network to learn how to identify and recognize the object of interest in an image. This allows the designed neural network to learn how to identify and recognize objects of interest in an image. Simple convolutional neural networks are built for image classification and the detection of single objects in images. A typical structure of a convolutional neural network is shown in the Figure 2.1.

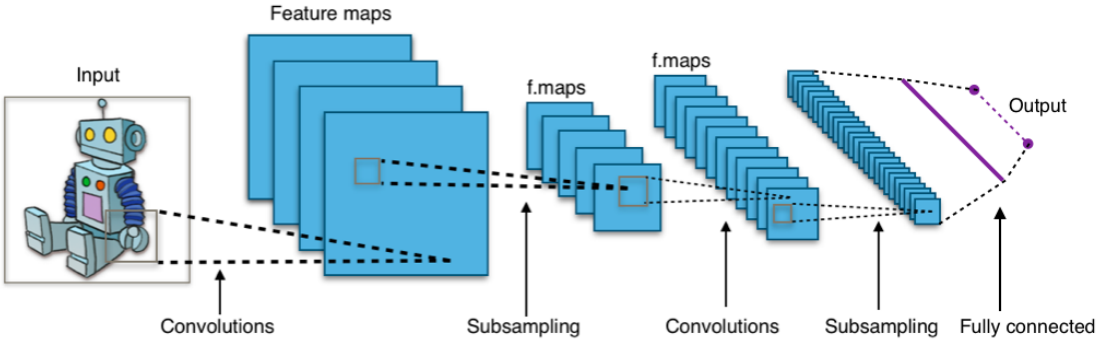


Figure 2.1: A typical structure of a convolutional neural network

2.1.2 Region-Based Convolutional Neural Networks

Girshick et al. [9] introduced a region-based CNN (R-CNN) for object detection. The pipeline consists of two stages. First, several object proposals are generated using selective search. These proposals are category-independent. Then, the image region within each proposal is warped to a fixed size and then mapped to a 4096-dimensional feature vector. This feature vector is then fed into a classifier and a regressor that refines the detection position. Object detection using R-CNN has the high accuracy of CNN for classification tasks. Its highlight is the transfer of the supervised pre-trained image representation used for image classification to object detection.

However, R-CNN has several disadvantages that cannot be ignored:

- The training process is a multi-stage pipeline. The R-CNN is first fine-tuned using log loss on a given region proposal. The third part is the regression of the learning detection frame.

- The training requires a lot of space and time, and the training process is prolonged because the features extracted by the convolutional neural network need to be written to disk during the training process, thus requiring a lot of physical storage space.
- The detection process is prolonged, and features are extracted from each target candidate frame in each test image during testing.

Faster R-CNN

To address the above shortcomings of R-CNN, the Faster R-CNN was proposed. It consists of two modules. As shown in Figure 2.2, it is a fully convolutional network for generating object proposals. This module is called the Regional Proposal Network (RPN). The output of this module is fed into the second module. The second module is the Fast R-CNN detector. It is used to refine the proposals. The key idea is to share the same convolutional layers for the RPN and Fast R-CNN detector up to their own fully connected layers. The images need to be fed to the CNN only once. It is used to produce and then refine object proposals. More importantly, because of the sharing of convolutional layers, it is possible to use a very deep network (e.g., VGG16 [10] and ResNet [11]) to generate high-quality object proposals.

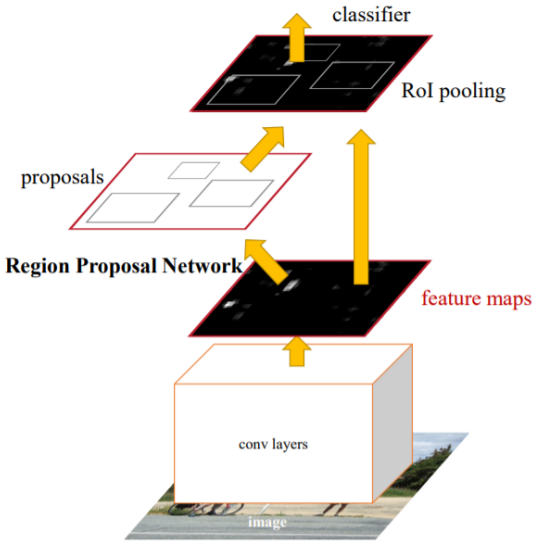


Figure 2.2: The structure of Faster R-CNN [1]

2.1.3 Mask R-CNN

Mask R-CNN [2] is a convolutional neural network (CNN) widely used for image segmentation. It detects objects in an image and generates a high-quality segmentation mask for each instance. This thesis uses Mask R-CNN to detect the cars in the exposed part of the floodwater and generate car masks.

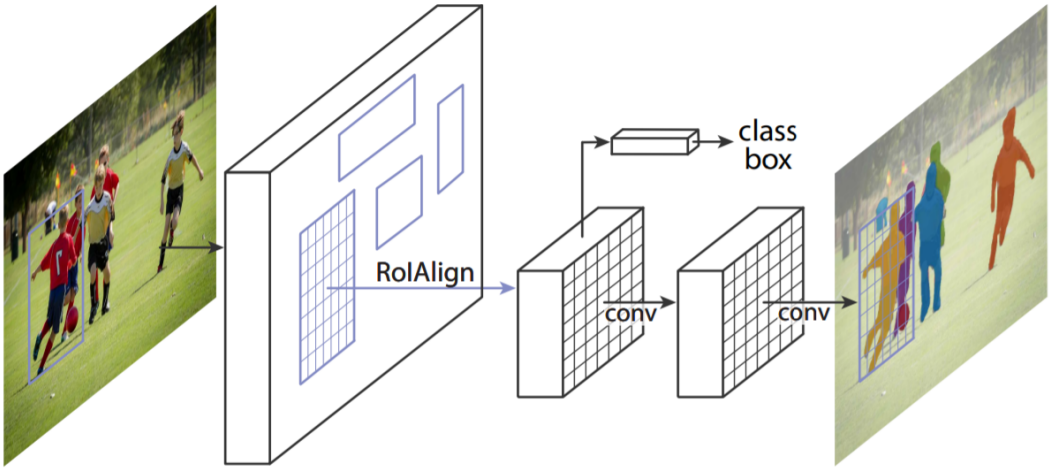


Figure 2.3: The Mask R-CNN framework for instance segmentation. [2]

The highlights of Mask R-CNN are as follows:

- Mask RCNN is a generic framework for instance segmentation.
- Add a mask branch, multi-task, to Faster R-CNN to achieve instance segmentation.
- This multi-task structure can not only be used for instance segmentation but also gives good results for human pose detection in the above paper.
- Use RoI Align instead of RoI pooling to improve the accuracy of instance segmentation.

Mask R-CNN adds a branch of the prediction mask in parallel to Faster R-CNN prediction frame and uses the mask branch to predict a binary mask at each RoI. The mask branch used to predict the mask is a small fully convolutional network that predicts the semantic mask at the pixel level for each RoI.

Since Faster R-CNN is not an algorithm designed mainly for pixel-level strength segmentation, especially in RoI pooling, only a coarse representation of the features is obtained. In order to correct the possible misalignment between pixels, the authors use RoIAlign instead of RoI pooling to obtain a significant improvement.

In addition, the authors decouple the prediction of categories and masks. Furthermore, predict a mask for each class independently, relying on the RoI classification branch of the network to predict classes.

2.1.4 Keypoint R-CNN

Keypoint R-CNN is an extension of Mask R-CNN. In the Mask R-CNN paper proposed by Keming et al., the framework of Mask R-CNN can easily be extended to Keypoint R-CNN. The keypoint's location is modeled as the one-hot mask, and Mask R-CNN is

adopt to predict K masks, one for each of K keypoint types. In this thesis, there are 66 keypoint types, so 66 masks need to be predicted.

The minor modifications are made to the segmentation system when adapting it for keypoints. Each keypoint is treated as an instance. After training, a one-hot $m \times m$ binary mask is output. Only a single pixel for one instance is labeled as foreground in this mask. During training, an m2-way softmax output is used to minimize the cross-entropy loss, which corresponds to a visible ground-truth keypoint. The keypoint head consists of a stack of eight 3×3 512-d conv layers, a deconv layer and 2× bilinear upscaling. It produces an output resolution of 56×56. A relatively high resolution output (compared to masks) is required for keypoint-level localization accuracy.

2.2 The Methods of Camera Pose Estimation

In this thesis, the purpose of performing camera pose estimation is to know the relative position and rotation of the camera in relation to the object. The pose information is used to project the 3D object onto the 2D image. If the 2D keypoints of the object and the corresponding 3D keypoints can be obtained, the pose information can be estimated by Perspective n-Point method (PnP) [12]. As shown in Figure 2.4, the similar triangular projection is the basis of PnP. Formally, PnP problem is defined as follows: Given a set of n 3D points P_i whose coordinates are known in some object coordinate frame O. A set of n 2D points p_i are the projections of the points P_i on the image plane I. C is denoted as the camera's center of perspective. Then, $\vec{v_i} = \widehat{Cp_i}$ is n directional vectors. So, the goal of the PnP problem is to estimate the rotation and translation of the camera in a world reference frame. To solve the problem, known 3D points and corresponding 2D image observations are required. Except for a few PnP methods such as DLT [13], which do not require camera intrinsic parameters input, all other methods require camera intrinsic parameters. In this thesis, the keypoints of each vehicle are obtained by Keypoint R-CNN. The PnP problem is constructed by labeling the keypoints on the 3D model with the 2D keypoints on the image. As follows, some common PnP methods are organized.

2.2.1 Direct Linear Transformation

Direct Linear Transformation (DLT) [13] maps points $x^W = [x, y, z, 1]^T$ in the world coordinate system to the image coordinate system $u = [u, v, 1]^T$. According to the projection relation, we can get Equation 2.1.

$$sU = KR_{cw}\left[I|-t^w\right]x^w = Px^w \tag{2.1}$$

Where, t^w denotes the coordinates of the camera's principal center in world coordinates, R_{cw} denotes the rotation relationship between the two coordinates, and K presents the

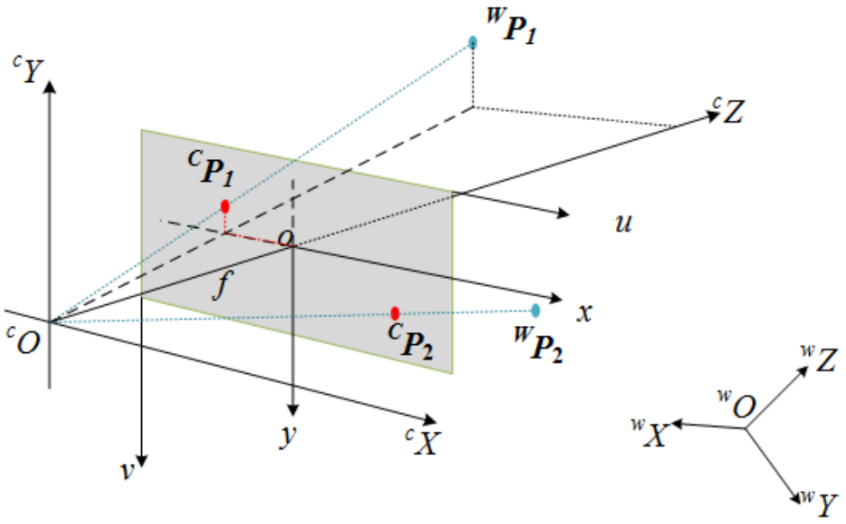


Figure 2.4: Similar triangles in the projection model [14]

camera intrinsic. DLT is an algorithm that computes the poses of the uncalibrated camera using greater than 6 matched points.

The specific derivation process is as follows:

$$\begin{aligned} sU_i = Px^w &= \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} x_i^w \\ &= \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix} x_i^w. \end{aligned} \tag{2.2}$$

So, there is,

$$s \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} p_1^T x_i^w \\ p_2^T x_i^w \\ p_3^T x_i^w \end{bmatrix}. \tag{2.3}$$

And,

$$\begin{aligned} u_i &= \frac{p_1^T x_i^w}{p_3^T x_i^w} \\ v_i &= \frac{p_2^T x_i^w}{p_3^T x_i^w}. \end{aligned} \tag{2.4}$$

After deformation we can get,

$$\begin{aligned} u_i p_3^T x_i^w - p_1^T x_i^w = 0 \\ v_i p_3^T x_i^w - p_2^T x_i^w = 0. \end{aligned} \tag{2.5}$$

So a linear equation can be constructed, where n denotes the number of matched point pairs.

$$\begin{bmatrix} -(x_1^w)^T & 0 & u_1(x_1^w)^T \\ 0 & -(x_1^w)^T & v_1(x_1^w)^T \\ -(x_2^w)^T & 0 & u_2(x_2^w)^T \\ 0 & -(x_2^w)^T & v_2(x_2^w)^T \\ \vdots & \vdots & \vdots \\ -(x_n^w)^T & 0 & u_n(x_n^w)^T \\ 0 & -(x_n^w)^T & v_n(x_n^w)^T \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = 0. \tag{2.6}$$

The above equation can be written as:

$$\underset{2n\times 12}{M}\ \underset{12\times 1}{p} = 0 \tag{2.7}$$

Due to the presence of error the above equation will not equal zero, the error is denoted as ω, so that,

$$\underset{2n\times 12}{M}\ \underset{12\times 1}{p} = 0 = \omega, \tag{2.8}$$

$$\Omega = \omega^T \omega. \tag{2.9}$$

Find p so that Ω obtains the minimum value which is,

$$\begin{aligned} \hat{p} &= \underset{p}{argmin}\, \omega^T \omega \\ &= \underset{p}{argmin}\, p^T M^T M p. \end{aligned} \tag{2.10}$$

The problem can be solved by using the singular value decomposition method (SVD), to perform singular value decomposition of the matrix m:

$$\underset{2n\times 12}{M} = \underset{2n\times 12}{U}\ \underset{12\times 12}{S}\ \underset{12\times 12}{V}^T \tag{2.11}$$

The matrix S has zero values except for the diagonal. Extract the value s_n on the diagonal, where $s_1 \leq s_2 \leq \ldots \leq s_{12}$. Take p as one of the column vectors v_i of the matrix V, which is the eigenvector corresponding to the singular value s_i. So to minimize Ω, choose $p = v_{12}$, which is the eigenvector corresponding to the minimum singular value s_{12}. This way we can solve for the matrix P. Decompose the matrix P to obtain the camera intrinsic K, R_{CW} and t^w.

2.2.2 The methods from OpenCV

In OpenCV [15], there are many PnP solvers to choose from. The SOLVEPNP_ITERATIVE and SOLVEPNP_EPNP methods are mentioned in this thesis.

Iterative method is based on Levenberg-Marquardt optimization [16]. In this case the pose with minimizes reprojection error is found, that is the sum of squared distances between the observed projections image points and the projected object points. When key points are not co-planar, at least 6 points are required. The initial values are found by DLT, and then iterative optimization is performed using Levenberg-Marquardt optimization.

Lepetit et al. proposed a new method, EPnP (Efficient Perspective n-Point Camera Pose Estimation), to solve PnP problem [17]. A solution in $O(n)$ for $n \geq 4$ is calculated. As with most PnP methods, EPnP is used to retrieve the locations of P_i relative to the camera coordinate frame. It is a simple task to retrieve the camera's orientation and translation so that the two sets of coordinates are aligned [18]. The key of the algorithm's efficient $O(n)$ implementation is, to represent the Pi as a weighted sum of $m \leq 4$ control points $C_1 \dots C_m$ and perform all further computation only on these points. For large n this yields a much smaller number of unknowns compared to other algorithms and therefore accelerates further computations.

2.3 Datasets

In this section the datasets covered in this thesis will be presented. These include the COCO dataset [3] used in MaskRCNN, the ApolloCar3D [4] dataset used in Keypoint R-CNN, and the ETHZ dataset [5] used to verify the flood depth.

2.3.1 COCO Dataset

The Mask R-CNN model used in this thesis is obtained by training with the COCO dataset. The full name of COCO is Common Objects in Context, a dataset provided by the Microsoft team that can be used for image recognition for object detection, segmentation, keypoint detection, adding captions, and more. The dataset is presented to advance the state-of-the-art in object recognition by placing the question of object recognition in the context of the broader question of scene understanding. Objects are labeled using per-instance segmentations to aid in precise object localization. The dataset contains photos of 91 object types, with 2.5 million labeled instances in 328,000 images. As shown in the figure, from left to right are image classification, object localization, semantic segmentation and instance segmentation.

Figure 2.5: COCO Dataset [3]

2.3.2 ApolloCar 3D Dataset

In this thesis, Keypoint R-CNN is used to detect the keypoints of the car. For this, the ApolloCar 3D dataset is used. ApolloCar 3D dataset is a dataset with 5,277 4K images. There are more than 60,000 car instances in these images. Otherwise, it also provides 79 industry-grade 3D CAD car models, the corresponding stereo images and accurate 2D keypoint annotations. Figure 2.6 shows that (a) is the original image and (b) is the

Figure 2.6: An example of ApolloCar 3D dataset [4]

keypoint annotation about each instance. The different positions of these keypoints have different categories. As shown in Figure 2.7, this dataset is illustrated for the key point annotation. The last figure shows the 3D model fitting results with labeled 2D keypoints.

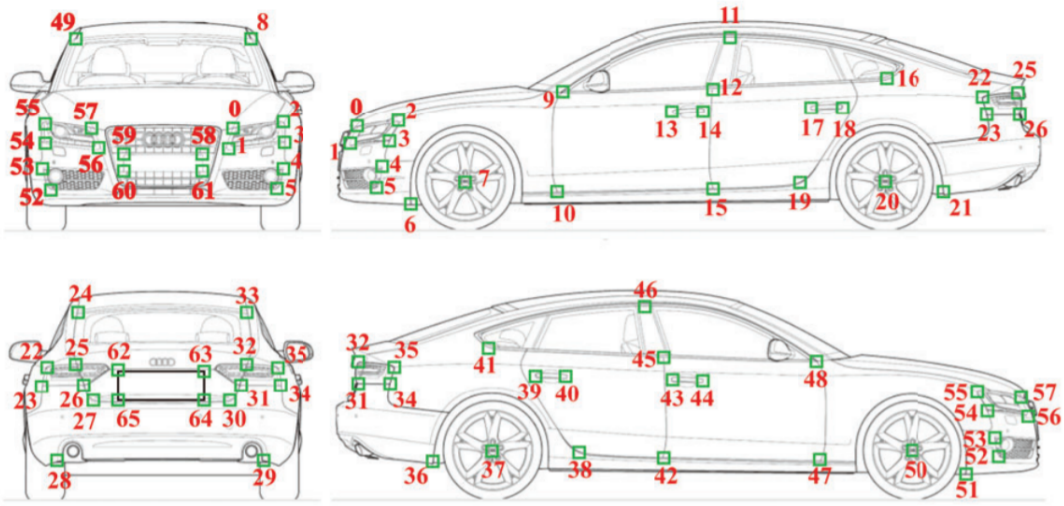


Figure 2.7: 3D keypoints definition for car models [4]

2.3.3 ETHZ Dataset

Chaudhary et al. manually labeled 7000 unique images, which were manually collected from various sources such as Google, Flickr, and National Geographic. Objects included in each image are from one or more of the five categories: person, car, bus, bicycle, and house. The Figure 2.8 shows that 11 levels are divided according to different depth ranges. As shown in Figure 2.8, the five categories of objects are divided into different areas according to their classes. The different classes are labeled against these flooded objects.

Level Name	Range	Value nearest integer
	cm	cm
level0	No water	0.0
level1	0.0 - 1.0	1.0
level2	1.0 - 10.0	10.0
level3	10.0 - 21.25	21.0
level4	21.25 - 42.5	43.0
level5	42.5 - 63.75	64.0
level6	63.75 - 85	85.0
level7	85.0 - 106.25	106.0
level8	106.25 - 127.5	128.0
level9	127.5 - 148.75	149.0
level10	148.75 - 170.0	170.0

Figure 2.8: Level to centimeters relation table [5]

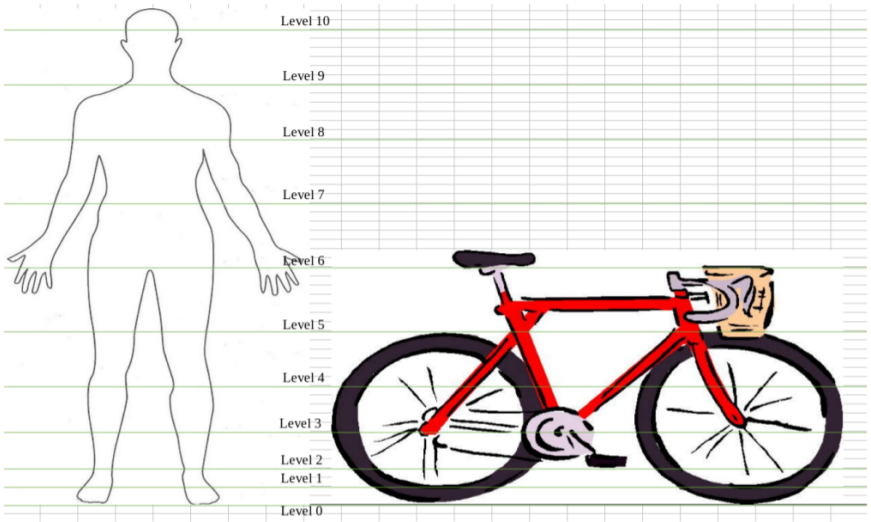


Figure 2.9: The annotation strategy for object [5]

2.4 Evaluation Metrics

To verify the class accuracy of the model output, the confusion matrix is used. In order to verify the performance of the method and improve the verification performance, TP,FP,FN, Precision and Recall are used. In the running phase of this method, the targets need to be matched and IoU is involved in the matching process. Next, these evaluation metrics are introduced.

First, when training the model, the class accuracy of the detected objects needs to be evaluated, and here the confusion matrix is used. The confusion matrix, also known as the error matrix [19], is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, both variants are found in the literature [20]. The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e. commonly mislabeling one as another).

To test the ability of the model to detect the target, Precision and Recall are used as evaluation metrics. First True Positive (TP), False Positive (FP), False Negative (FN), True Negative (TN) will be presented. Figure 2.10 shows that the results are divided into predicted and actual. Corresponding to the graph, if the prediction is positive and the actual is positive, it is TP. Similarly, FN, FP, and TN.

When TP,FP,FN are known, Precision and Recall can be calculated by the following equations.

$$Precision = \frac{TP}{TP + FP} \tag{2.12}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.13}$$

Therefore, *Precision* mainly evaluates the false detections of the results and *Recall* is used to evaluate the leakage of the results.

		Predict	
		P	N
Actual	P	TP	FN
	N	FP	TN

Figure 2.10: Confusion Matrix

Intersection over Union (IoU) is an evaluation metric used to measure the accuracy of an object detector. In addition to this, IoU is also used for target matching. If the bounding

box of two objects, $B1$ and $B2$, is known, the area where $B1$ and $B2$ intersect is A_I. The union area of B1 and B2 is A_U. The IoU is calculated by Equation 2.14.

$$IoU = \frac{A_I}{A_U} \tag{2.14}$$

3 Related Work

In this chapter, related work in the thesis is presented. The need to estimate and measure floods is becoming more and more urgent due to the significant threat they pose to human production, life, and property. With the development of artificial intelligence, many studies are now based on deep learning. There are currently some methods that output water level estimates directly through deep learning. For example, Chaudhary et al. [5] add a head for water level estimation by changing the Mask R-CNN framework. This is an end-to-end approach where the results are output through a single model. However, a large dataset is used to regress the parameters of the model by this approach. The model cannot learn the 3D information of the object and the spatial relationships, but can only estimate by the features of the image. Therefore, this method is not accurate enough. The method of water level estimation presented in this thesis is by detecting cars that are submerged. Next, the existing methods are reviewed by classifying the submerged objects.

3.1 Object based Methods

The dominant approach is to estimate the water depth from submerged objects. These objects are usually of a fixed shape and size and are more common in floods. The objects used in the current study are more often pedestrians or cars.

3.1.1 Human Based Methods

Since pedestrians are often present in flood scenarios, many studies have used pedestrians to estimate water depth. Geetha et al. [21] used the crowds in flood to calculate the extent of the flood. A color-based segmentation is used to segment out the water. The brown color intensity of the water part, along with its depth with proportion to humans detected, is then analyzed to estimate the extent of the flood. Meng et al. [22] detected and segmented human objects from inundated images to infer flood depth. They extracted the keypoints of humans by Mask R-CNN. The ratio of inundation is calculated by determining the key points exposed to the water, and thus the water depth is calculated. Feng et al. [23] also extracted the key points of the pedestrian by Mask R-CNN. Then, the images were classified according to different submerged body parts, i.e., ankles, knees, hips, and chest. Finally, the images containing people were classified into four severity levels, corresponding to different water levels.

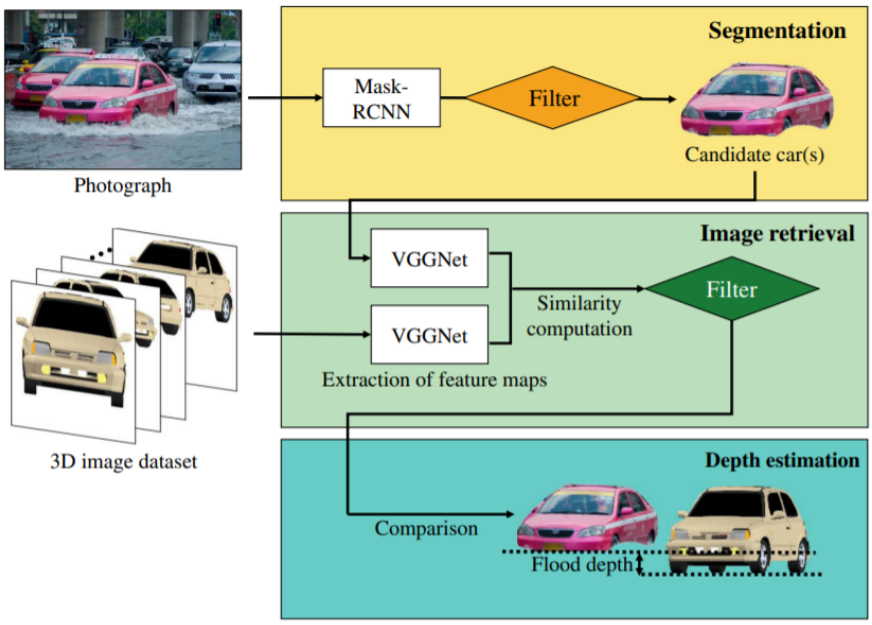


Figure 3.1: Water Depth Detection by Vehicles [24]

3.1.2 Vehicle Based Methods

In addition to pedestrians, cars are often used as a reference for flood water depth estimation. Park et al. [24] used a deep learning-based object segmentation network Mask R-CNN to detect vehicular objects in images. As shown in Figure 3.1, the segmented image is then found with the most similar 3D rendered image by VGGNet [25]. After comparing the two, the depth of the flood is calculated by the wheels. Besides, Sazara et al. [7] proposed mixed-use image processing and deep learning methods to calculate the water depth in flood images. The vehicle tires are extracted by the deep learning method, and the water depth is calculated by calculating the proportion of the vehicle tires in the water after processing the tires by the traditional computer vision method.

3.1.3 Traffic Sign Based Method

Other objects have also been used in current studies as references for flood water depth estimates. Kharazi et al. [26] calculated the depth of the flood by detecting the stop signs on the roadside. They are based on the octagonal shape of the stop signs with standardized equal height and width. As shown in Figure 3.2, the depth of flooding can be calculated by comparing it with the previously photographed non-flooded signs by scale.

3.2 Overview

Each of these studies has its strengths and weaknesses—for example, pedestrians as a reference for flood water depth estimation. The probability of pedestrians appearing in

a flood scenario is relatively high, making the success rate of flood estimation high. The detection of pedestrians plays a positive role in rescue in addition to estimating water depth. However, these studies also reveal many shortcomings. For example, the height of pedestrians varies greatly from pedestrian to pedestrian, and the posture of pedestrians in the water is difficult to predict. All these factors will lead to errors in water depth estimation.

The advantage of using a car as a reference is that the size and category of the car are relatively fixed, allowing more accurate results to be obtained. As mentioned above, most studies are based on the submerged wheels in estimating the depth of the flood. These methods have a high degree of constraint. These methods will fail if the flood depth is too high, causing the wheels to be submerged.

The advantages and disadvantages mentioned above are analysed. Advantage of the vehicle's relatively fixed size and type is taken to propose a method for flood estimation based on the whole vehicle. Thus, the constraint of using wheels for estimation is effectively avoided.

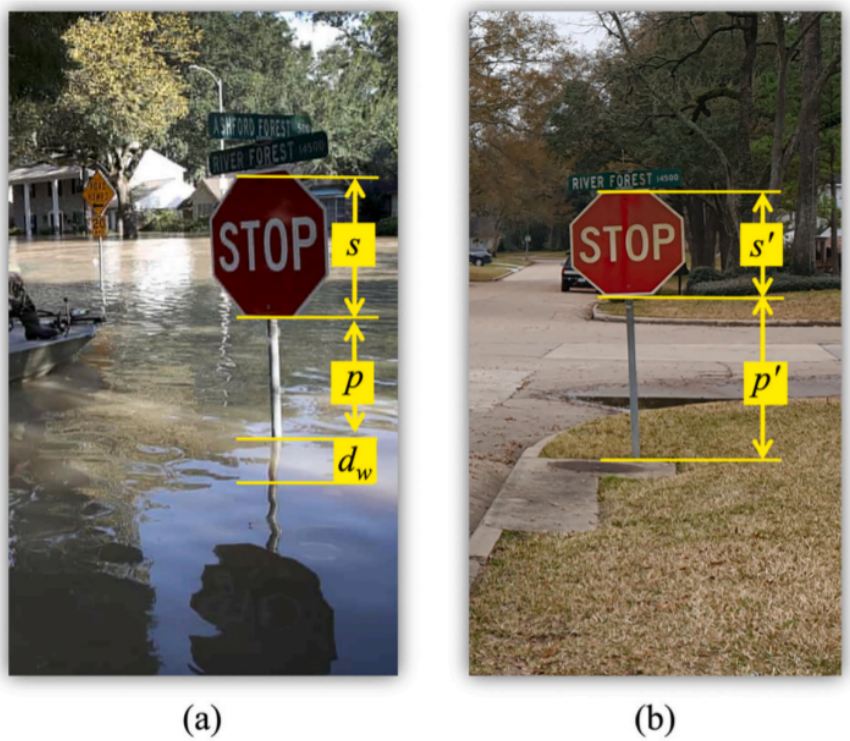


Figure 3.2: Water Depth Detection by Stop Signs [26]

4 Methodology

The method proposed in this thesis is described in detail in this chapter. Figure 4.1 shows the main framework. The framework is divided into three parts, extract submerged vehicles from images, pose estimation and water level estimation.

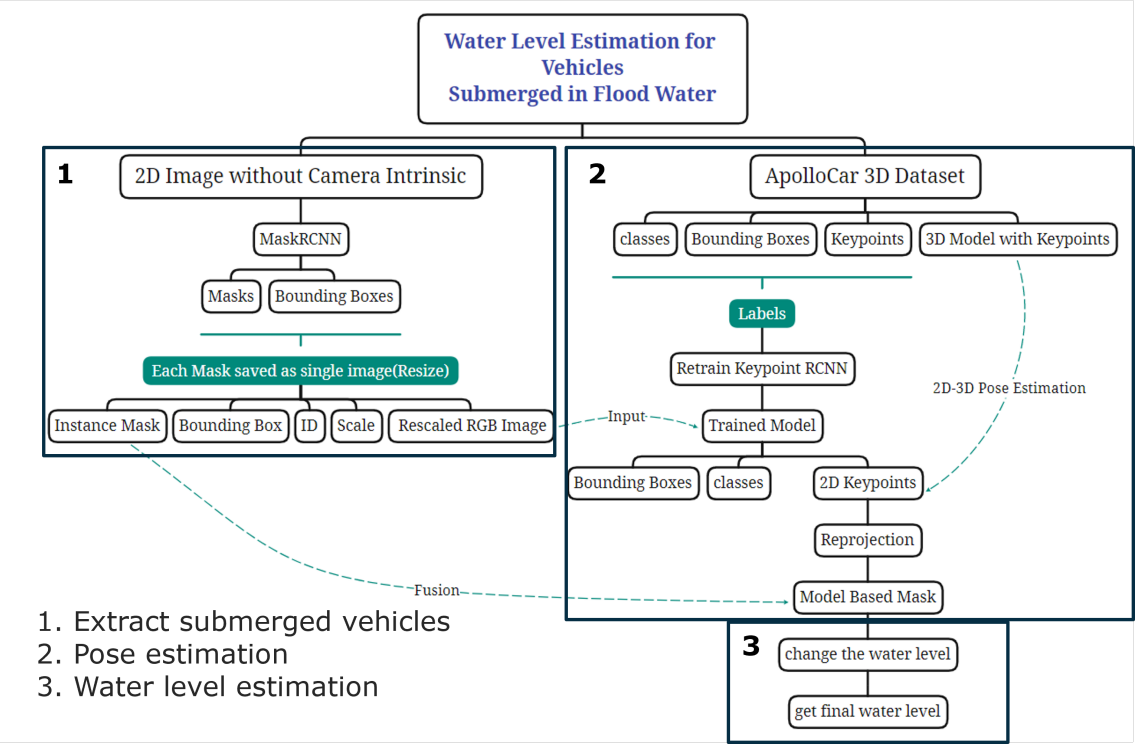


Figure 4.1: Framework of the proposed method

4.1 Extract submerged vehicles from images

Mask R-CNN [2] is used to extract the cars exposed to the water in this section. The Mask R-CNN model is pre-trained on COCO dataset [3] and can detect 80 different objects. As shown in Figure 4.2, the Mask R-CNN model can output the information of class, bounding box, mask, etc. In this thesis, only the cars that appear in the image need to be detected.

Figure 4.2: Output of Mask R-CNN

Therefore, the results of Mask R-CNN need to be filtered. After filtering, only the targets with the class of car are retained.

Each car instance in all images has its own fixed ID. Each ID contains the name of the image where the instance is located, the location, width, and height of the instance (bounding box). Finally, as shown in Figure 4.3, each instance and its corresponding mask are extracted according to the bounding box.

Figure 4.3: Extraction of the instance

4.2 Pose Estimation

Once each car instance has been extracted, the 3D model needs to be projected onto the 2D image to fit these car instances. The purpose of pose estimation is that the 3D model can be reprojected onto the 2D image through the pose information between the camera and the 3D model, and the reprojected car mask is generated. 3D models are already

available for us in the ApolloCar 3D dataset. In this section, we focus on estimating the poses between the 3D model and the camera.

4.2.1 Methods of Pose Estimation

To obtain the pose information, the following three methods are tried.

- Deep learning-based method to directly obtain the object's pose information in the 3D coordinate system - GSNet [27].
- Deep learning-based method to detect the object's center position in the image coordinate system, depth information, and rotation information in the 3D coordinate system - CenterNet.
- Traditional computer vision-based method to obtain the pose information - PnP.

GSNet

GSNet (Geometric and Scene-aware Network) [27] is an end-to-end framework. Through the neural network, 6 DoF poses are estimated. The pose information is used to reconstruct the city street scene from monocular images by 3D car models. GSNet extracts feature through Keypoint R-CNN and fuses the features to directly regress the 6DoF pose and shape in one forward process.

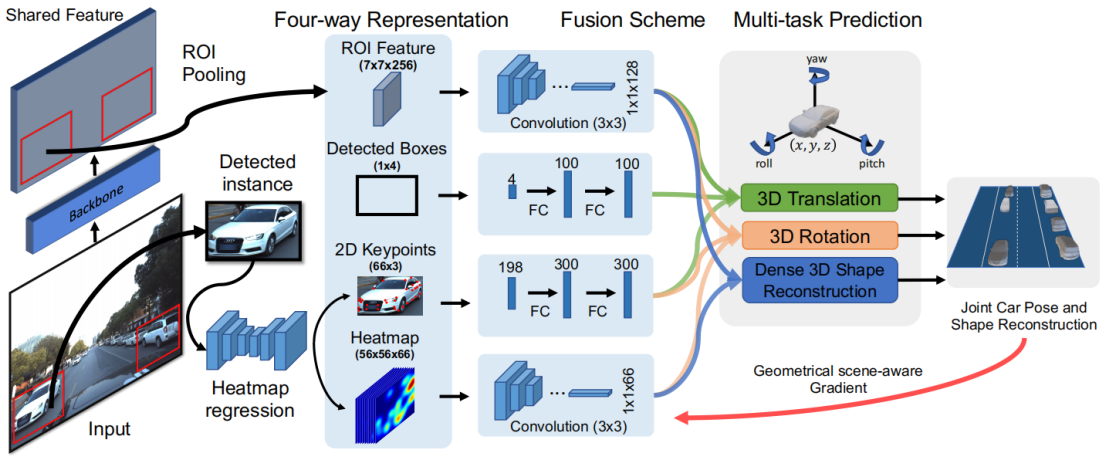


Figure 4.4: Framework of GSNet [27]

Figure 4.4 shows the framework of GSNet. The images are fed into the model and the first pass through the Keypoint R-CNN to obtain RoI features, detected boxes, 2D keypoints, and heatmap. These results are combined in several layers of the convolutional network to output parameters for translation, rotation, and 3D Shape Reconstruction.

Figure 4.5 shows the results generated by feeding the images from the ApolloCar 3D dataset into GSNet. It can be seen that the fit is not good. The reason is that the network

Figure 4.5: Output of GSNet [4]

generates the positional information in the world coordinate system directly through the model. Once this information is biased, it will cause a bias in the position projected onto the 2D image. Also, it is experimentally verified that the images input to the GSNet network needs to know the camera intrinsic parameters in advance. If the camera intrinsic parameters are unknown, it will cause the model to fail to output the correct result. Therefore, this method does not apply to our method.

CenterNet

Since the proposed method needs to fit the 3D car model to the 2D car accurately, a new method need to be designed. By finding the center point of the car in the image coordinate system, the depth information, and the rotation information, the 3D model can be accurately projected onto the image. CenterNet is an excellent way to achieve this requirement.

CenterNet represents objects by a single point at their bounding box center. Other properties, such as object size, dimension, 3D extent, orientation, and pose, are then regressed directly from image features at the center location. Object detection is then a standard keypoint estimation problem. The input image is fed to a fully convolutional network that generates a heatmap. Peaks in this heatmap correspond to object centers. Image features at each peak predict the objects bounding box height and weight.

3D detection of CenterNet estimates a three-dimensional bounding box per objects and requires three additional attributes per center point: depth, 3D dimension, and orientation. In our attempt, ResNet50 is used as the backbone network of CenterNet.

Figure 4.6: Output of CenterNet [4]

From Figure 4.6, we can see that the fitted car position is more accurate compared with the first method, but the depth information obtained by this method is not accurate, resulting in a large or small fitted car. This result will also affect the final flood depth estimation.

PnP

Inspired by GSNet, if Keypoint R-CNN can obtain each vehicle's class and keypoint information, and the 3D model keypoints of the corresponding class can also be obtained, it becomes a 2D-3D pose estimation problem. A typical method to solve such a problem is PnP. This is a method of traditional computer vision.

However, as introduced in Chapter 2, PnP relies on the camera intrinsic parameters for most of the solutions to the PnP problem, except for methods such as DLT. Furthermore, as introduced in Chapter 2, for method DLT, noise-free 2D keypoints and corresponding 3D keypoints are required. Since the 3D model and the car on the image cannot be matched precisely, and the keypoints cannot be made entirely accurate, the noise-free requirement is not satisfied. For this reason, the DLT method is used to verify whether noisy keypoints can be used. As shown in Figure 4.7, the red points are the vertices of the

3D model. If the correct positional relationship is obtained, the vertices will be projected in the white mask area. It is obvious that the result of reprojecting the 3D model vertices onto the 2D image using the DLT solution is entirely wrong.

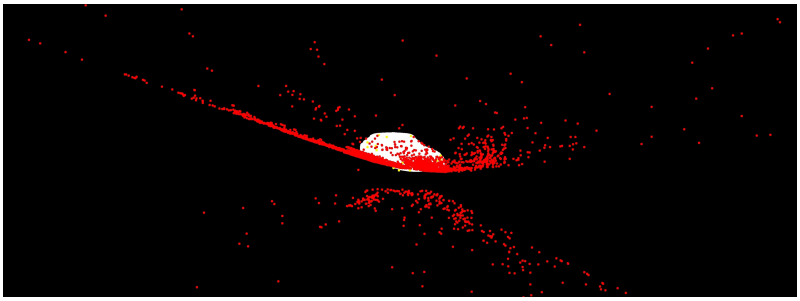


Figure 4.7: Test of DLT method

The main parts of the camera intrinsic parameters are the focal lengths of the x and y axes denoted as f_x and f_y, and the coordinates of the main point of the camera, denoted as $(c_x,\ c_y)$. The magnitude of the focal length affects the scale of the points in the image coordinate system to the points in the corresponding world coordinate system. On the other hand, the principal point coordinates affect the offset relationship between the image origin and the projection center. When this information cannot be obtained, it is impossible to get the object's exact position in the world coordinate system. In order to solve the problem of not being able to get the camera intrinsic parameters for every image, the fake camera intrinsic parameters are needed. If the size of the input image is the same as the size of the image in the Apollo dataset, the size and position of the object in the input image are also close to the size and position of the object in the Apollo dataset. Then these fake camera intrinsic parameters can be the same as the camera intrinsic parameters in the Apollo dataset.

The translation and rotation vectors obtained by the PnP solver deviate from the actual ones if fake camera intrinsic parameters are used. However, the ultimate goal of this paper is not to obtain the natural positional relations but to correctly project the 3D model onto the 2D image using the positional relations. Therefore, the 3D model can accurately be projected onto the image using the fake camera intrinsic parameters and the pose information obtained by the fake camera intrinsic parameters.

The use of PnP to solve flood water level estimation has been proven through testing to be the most effective of the available solutions. In the next sections, the method is developed based on PnP.

4.2.2 Training of Keypoint R-CNN

In order to enable the PnP solver to derive more accurate results, the accuracy of 2D key points needs to be improved. The trained Keypoint R-CNN model of GSNet is used as a pre-trained model. Since the ApolloCar 3D dataset used by GSNet has a single scene,

it does not work well for car detection and keypoint extraction in flood scenes. This is because a car in flood will be submerged under the floodwater. Furthermore, the absence of this part of the features will cause the neural network to fail to detect the car accurately. Therefore a series of fine-tuning on top of the original model is needed.

There are three main steps for fine-tuning. First, 79 classes in the original model are changed to 3 classes, i.e., different kinds of cars are classified into three classes, Sedan, Hatchback, and SUV. The model is fine-tuned so that the model can extract the features of these three categories well. Then, the model needs to be fine-tuned to be adapted to different resolution input images. The last fine-tuning enables the model to improve the detection of submerged cars.

In the first step, it can be demonstrated through the experiments in Chapter 5 that translating 79 classes of models into three classes can yield better class accuracy. As shown in Figure 4.8, ApolloCar 3D provides 79 different car models, and each model is converted into the corresponding three classes. After updating the annotation file, the model is fine-tuned. The training is stopped when the class accuracy of the model stabilizes. After the training, the new model is obtained, denoted as $M1$.

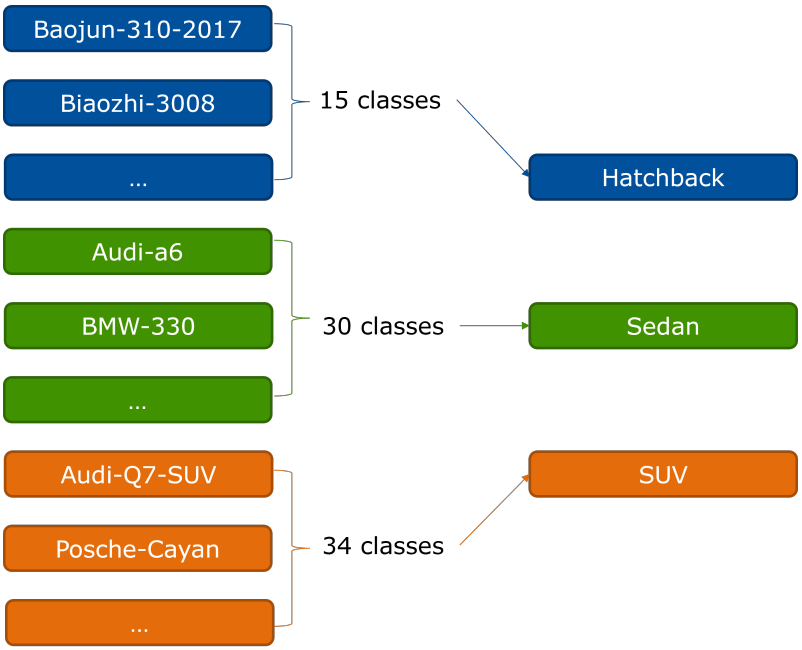


Figure 4.8: Translation of classes

In the second step, the dataset is first required to be processed in order to make the model better adaptable to images of different resolutions. The Apollo dataset is processed by mean filtering. The window size of the mean filtering is set randomly. In addition to this, 200 images are randomly selected and re-mean filtered as the test set for the experiments in Chapter 5. We refer to the new dataset as ApolloCar 3D Blurred dataset. Figure 4.9 shows the comparison of the original image and blurred image from the new dataset. $M1$

is used as a pre-trained model, and ApolloCar 3D Blurred dataset to fine-tune the model and generate a new model, denoted as $M2$.

(a) Original image in ApolloCar 3D dataset (b) Blurred image in ApolloCar 3D dataset

Figure 4.9: the comparison of the original image and blurred image [4]

The third step is mainly to improve the detection rate of submerged vehicles. For this, the ApolloCar 3D Flurred dataset is processed to make it more similar to the flooding scenario. The new dataset is generated as follows.

- Read each car's class information from the training set annotation file of ApolloCar 3D, and get the 3D model of the corresponding class according to the class information.
- The file of the 3D model is composed of vertices and triangle information. The floating-point value from the lowest point to the middle point of the 3D vertices is randomly selected as the threshold. As shown in Figure 4.10, the points below the threshold are filtered out, and the 3D model is reacquired.
- Based on the positional information of each car in the annotation file and the camera intrinsic, the corresponding 3D model is reprojected onto the 2D image to generate a mask for each car. As shown in Figure 4.11, the left image is the original image, and the right image is the mask generated by the submerged model after reprojection.
- Randomly selected flood scenes are used as the background of the new dataset, as shown in left image of Figure 4.12.
- Extract the RGB information of the blurred dataset within the mask and transfer it to the flood background, as shown in right image of Figure 4.12.

This new dataset is called ApolloCar 3D Flood, and this dataset is used to fine-tune $M2$ to generate the model $M3$.

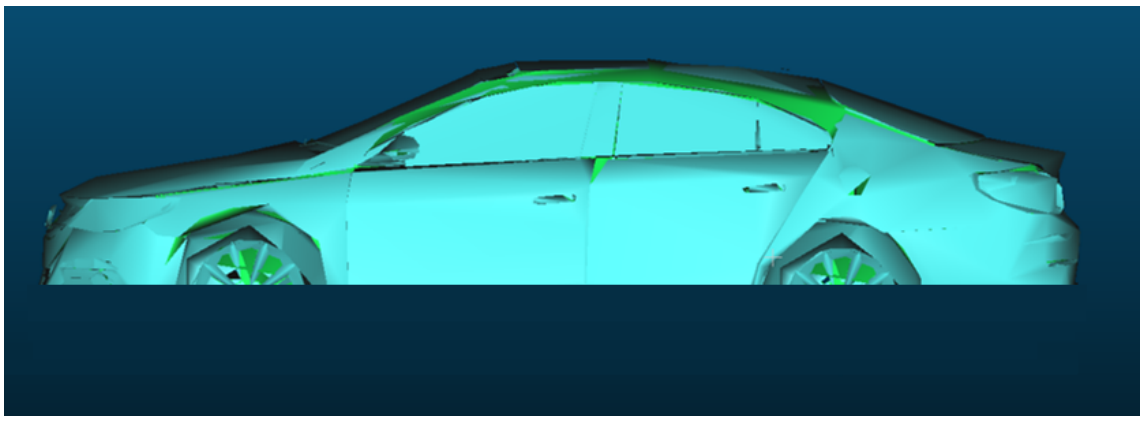


Figure 4.10: Simulation of submerged model

Figure 4.11: Masks of flooded model [4]

4.2.3 Images Input

ETHZ dataset is used for testing. This is the flood scene dataset, 100 images with submerged cars are selected from this dataset. In this dataset, the size of the images varies. If these images are directly input into Keypoint R-CNN to detect the keypoint information on the car, it is impossible to get good results. In Keypoint R-CNN network, the size of the input image is 3384*2710. Also, in order to make the camera intrinsic parameters of the input image closer to the camera intrinsic parameters of the Apollo dataset, it is necessary to resize the image to the specified size. If resizing is performed directly, it will face many problems. One of them is that the original image has different ratios of width and height, which will change the original ratio after resizing and distorting the car. Keypoint R-CNN is a convolutional neural network that cannot obtain global information. The keypoint is a point with specific features. If the area of the keypoint is too large or too small, the model cannot extract the feature information. Therefore, the strategy we adopt is to resize the bounding box of each car extracted by Mask R-CNN to a suitable size and place it into the background of 3384*2710 individually.

Figure 4.12: Simulation of flooding scenario

In the next chapter, the experiment of choosing the optimal width is described. After the experiments, the width of each car instance is determined to be 600 pixels. To make the resized image return to the size of the original image, the scale of each instance before and after resizing need to be recorded.

Assuming that s is used to present the scale, the width and height of the bounding box are presented as w and h. The width and height after resizing are presented as w_r and h_r. Then s and h_r can be obtained that $s = w/w_r$ and $h_r = h/s$. After experiment, w_R is set to 699. Figure 4.13 shows the resized image and the resized mask of one of the instances in the input image. Figure 4.2 shows that the resized image will be used as the input image of Keypoint R-CNN.

Figure 4.13: Instance of Mask R-CNN Output

4.2.4 Keypoint Filtering

The images are fed into the trained $M3$ model, which outputs information about the car's bounding box, class, and keypoints. Figure 4.14 shows the output of Keypoint R-CNN.

From the figure, it can be seen that a car instance will have multiple output bounding boxes. The reason is that the output is not processed by non-extreme suppression (NMS). The purpose of doing so is to keep all the outputs as much as possible and find the optimal output through the algorithm. The results of each bounding box are shown separately in Figure 4.14. Each bounding box will act as a candidate. By comparing the keypoints of each bounding box with the labeling rules in Figure 2.7, it can be seen that many of the keypoints are wrong. Therefore, an algorithm is needed to filter out the wrong keypoints.

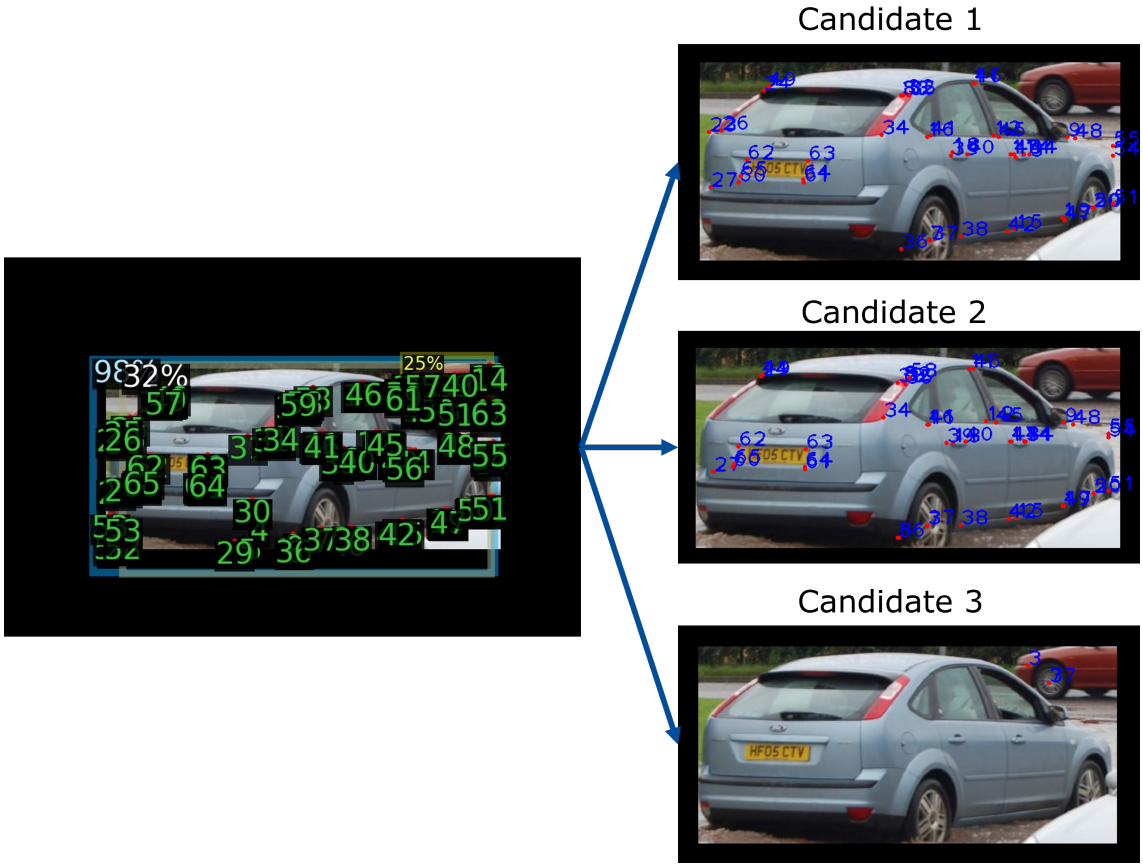


Figure 4.14: Output of Keypoint R-CNN

(a) Before filtering

(a) After filtering

Figure 4.15: Keypoints before and after filtering

In Figure 4.16, it can be seen that there are 18 keypoints in the front and back and 25 keypoints on the left and right sides. As shown by the red circle in Figure 4.16, 10 keypoints on the left and right sides are repeated with the key points in the front and back, such as 0, 1, 2, 3, 4, 5. The duplicate keypoints on the left and right sides are eliminated, resulting in 15 keypoints on each side. Next, the keypoints of each currently detected instance are counted. Calculate how many of these keypoints belong to the front, back, left, and right sides. For example, the number of keypoints belonging to the back is greater than the front, and the number of key points belonging to the right is greater than the left. Assuming that the car in the image is facing us from the back right side. Finally, filter out the keypoints that do not belong to these two sides. Figure 4.15 shows the information of keypoints before and after filtering.

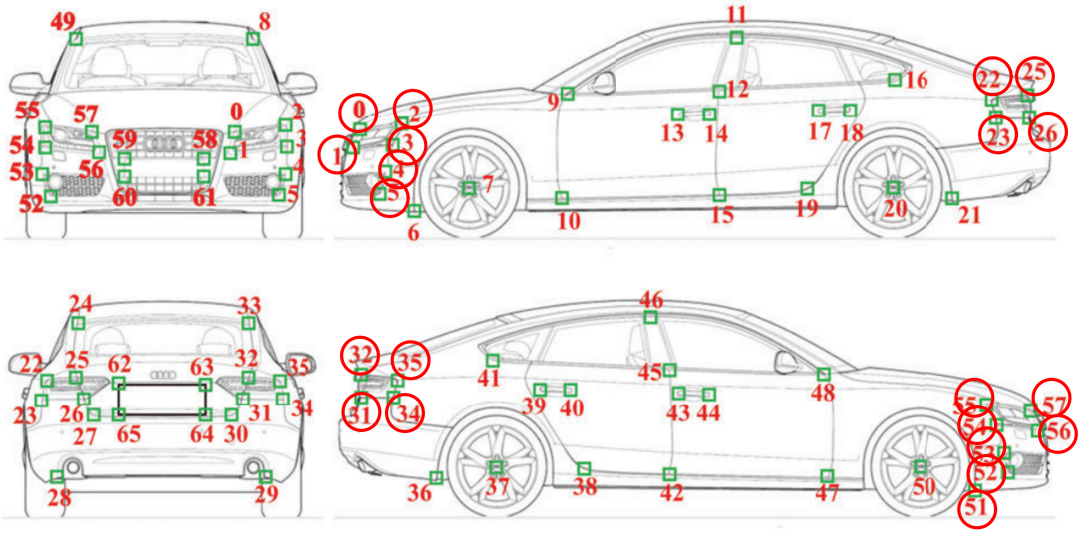


Figure 4.16: Selection of keypoints for each side

4.2.5 Keypoints on the 3D model

After obtaining the keypoints on the 2D image by the Keypoint R-CNN model, the corresponding keypoints on the 3D model are also required. Since the ApolloCar 3D dataset only provides us with 3D models of 79 different cars, there is no information on keypoints. Therefore, manual annotation is required to get the keypoints of the 3D models.

In order to save time and labor costs, one type of car belonging to Sedan, Hatchback, and SUV is selected, respectively, for labeling. This is the reason why the original 79 classes is converted to 3 classes when training Keypoint R-CNN. The car type of dazhong is chosen for Sedan, boshijie-kayan is chosen for SUV, and biyadi-2x-F0 is chosen for Hatchback. Figure 4.17 shows the biyadi-2x-F0 model annotated with CloudCompare software.

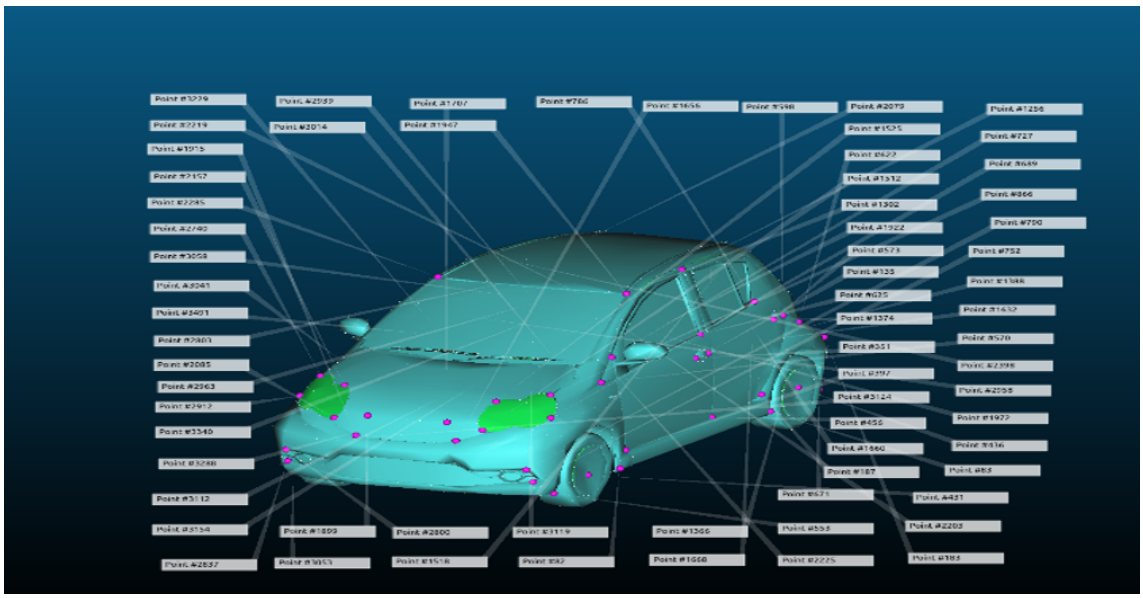


Figure 4.17: Annotation of 3D keypoints

4.2.6 PnP Solver

After the above steps, the 2D keypoints and the corresponding 3D keypoints are obtained. At this stage, the problem of solving the pose information can be converted into a PnP problem. The specific algorithm is shown in Algorithm 1.

As shown in Algorithm 1, I and M are the images and masks generated by Mask R-CNN after resizing through all the images. First, iterate through each resized image and mask. Initialize the maximum IoU to zero, the final results res is set to the empty and the camera intrinsic parameters are set to be the same as that of the ApolloCar 3D dataset. After feeding the images into the Keypoint R-CNN, each car instance generates many bounding boxes, which are called candidates, and then iterate through each candidate to obtain the keypoints and class information of the candidate. The keypoints are filtered by the method mentioned in 4.2.4, and the filtered keypoints result is obtained, denoted as K_{2d}. Select the corresponding 3D model according to the class information. Then read the 3D model vertices, triangles, and coordinates of the corresponding 3D key points K_{3d}. After initializing the minimum reprojection error E_{min} to positive infinity, 200 iterations are performed. In each iteration, 12 keypoints and the corresponding 3D keypoints are randomly selected, and input to the PnP solver with the fake camera intrinsic parameters, and the rotation vector r and translation vector t are obtained. Based on r and t, K_{3D} is projected onto the image. The error between the projected points and the keypoints on the 2D image is calculated. The reprojection error E_{reproj} is calculated by 4.1. Compare E_{reproj} and E_{min}. If E_{reproj} is less than E_{min}, update E_{min}. and save r, t and c to r_{best}, t_{best} and c_{best}. After 200 iterations, determine if the final E_{min} is less than the threshold T_{error}. If less, project the 3D model onto the image to generate mask, M_{rep}. The threshold T_{error} can be set manually. The smaller T_{error}, the stricter the filter condition of r and

t. Next, the IoU of M_{rep} and M_n is calculated. If IoU is greater than IoU_{max}, IoU_{max} is updated, and r_{best}, t_{best} and c_{best} are recorded in r_{out}, t_{out} and c_{out}. After iterating through all the candidates, select the candidate with the largest IoU and add r_{out} and t_{out} to the set res if greater than the set threshold T_{iou}. As shown in Figure 4.18, assume an instance has 6 candidates. By calculation, the candidate with the largest IoU between the mask of the exposed part of the instance and the mask generated by reprojection is selected.

$$E_{reproj} = \sum_{i=0}^{n} \| K_{3drep,i} - K_{2d,i} \|_2 \tag{4.1}$$

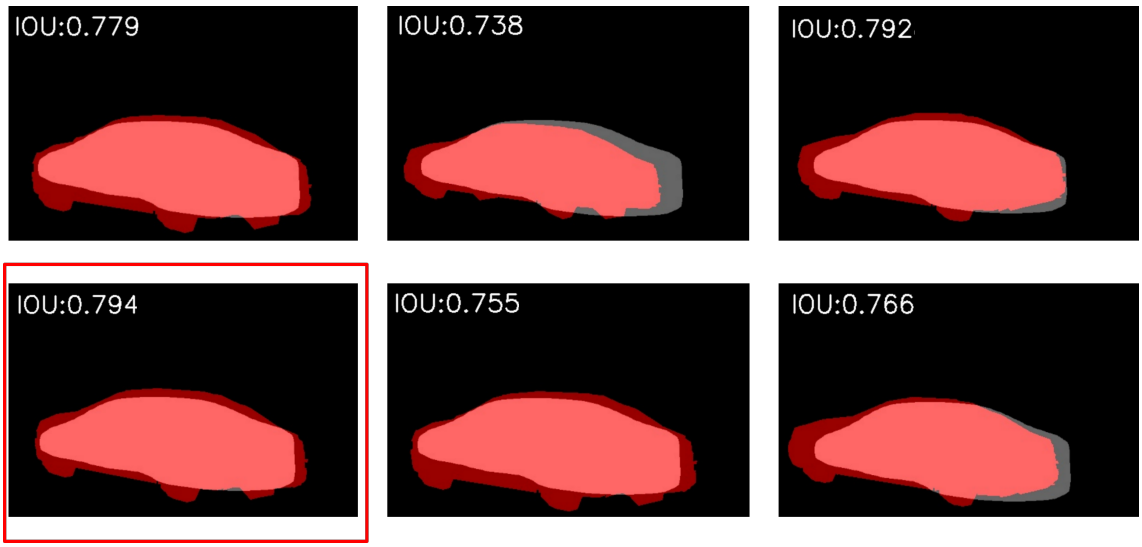


Figure 4.18: Selection of the candidate with max IoU

In this thesis, the package solvePnP from OpenCV is used. $SOLVEPNP\_EPNP$ and $SOLVEPNP\_ITERATIVE$ are used to try them out, respectively. It is verified experimentally that $SOLVEPNP\_ITERATIVE$ works better. It can also be seen from Algorithm 1 that the method is similar to RANSAC (Random Sample Consensus) [28]. We also use the package of SolvePnpRansac in OpenCV for the experiments, and the results show that the method proposed in this thesis has better results. In addition, other attempts at randomly selecting keypoints are made. Keypoints are selected randomly according to the two sides of the car facing us separately. The number of keypoints is chosen randomly in proportion to the number of keypoints belonging to both sides. Experiments show that random selection of keypoints without following the sides gives better results. The experimental results will be described in detail in the next chapter.

4.3 Water Level Estimation

The mask M_n, rotation vector r, translation vector t, and class information c of each instance can be obtained from the above sections. In this section, the algorithm is designed for water level estimation based on these known quantities.

As shown in Algorithm 2, iterate through every the image I_n, mask M_n, and the corresponding res_n of the car instances. First initialize n and IoU_{last} to 0, and then perform 100 iterations. In each iteration, set the flood water level d_{flood} to $0.02 * n$ and get its corresponding *vertices* and *triangles* according to the information of cls. As shown in Figure B, since the origin coordinates under the world coordinate system are the center of the 3D model, the height of the *vertices* submerged by the current water level should be less than $-(v_{bottom} - d_{flood})$. Set these vertices to the water surface position, i.e. $-(v_{bottom} - d_{flood})$. Then, new vertices represent the vertices information of the 3D model submerged by the current water level. If the current IoU is greater than the last IoU, the current water level can continue to rise. If it is less than, it means that the current water level has reached the actual water surface, then the water level information at this time is the final output water level information.

Algorithm 1 Calculate Pose infomation r and t

Input: Input images, I, and masks generated from Mask R-CNN, M. Fake camera intrinsic parameters, K. Reprojection error threshold, T_{error}. IoU threshold, T_{iou}.

Output: The result set of every images, res.

for every car instance image I_n and mask M_n in images I and masks M **do**
    initialize $IoU_{max} = 0$, the final result $res = \emptyset$, camera intrinsic K
    for every candidate attributes i_m **do**
        obtain keypoints of original 2D image from i_m, K_{ori2d}
        obtain keypoints after filtering, k_{2d}
        obtain class of the instance c from i_m
        if $c = 0$ **then**
            obtain vertices and triangles of $biyadi - 2x - F0$
        if $c = 1$ **then**
            obtain vertices and triangles of $dazhong$
        if $c = 2$ **then**
            obtain vertices and triangles of $boshijie - kayan$
        obtain corresponding keypoints of k_{2d} in 3D, K_{3d}
        initialize $n = 0$, $E_{min} = inf$
        while $n < 200$ **do**
            $n = n + 1$
            select 12 keypoints from k_{2d} randomly, K_{2dsel}
            obtain corresponding 3D keypoints of K_{2dsel}, K_{3dsel}
            Calculate t and r using PnP
            reproject K_{3d} to 2D image using t and r, K_{3drep}
            calculate reprojection error between K_{3drep} and k_{2d}, E_{reproj}
            if $E_{reproj} < E_{min}$ **then**
                $E_{min} = E_{kpt}$
                $r_{best} = r$
                $t_{best} = t$
                $c_{best} = c$
        if $E_{min} < T_{error}$ **then**
            reproject 3D model to 2D image using r_{best} and t_{best}, generate the mask, M_{rep}
            calculate the IoU of M_{rep} and M_n
            if $IoU > IoU_{max}$ **then**
                $IoU_{max} = IoU$
                $r_{out} = r_{best}$
                $t_{out} = t_{best}$
                $c_{out} = c_{best}$
    if $IoU_{max} > T_{iou}$ **then**
        add r_{out}, t_{out}, c_{out} to res

Algorithm 2 Water level estimation

Input: Input images, I, and masks generated from Mask R-CNN, M. The result set from last section, res.

Output: Water level of each image, d_{flood}

for every car instance image I_n, mask M_n and result res_n in images I, masks M and the result set res **do**
    initialize $IoU_{last} = 0$
    while $n < 100$ **do**
        set flood water level d_{flood} to $0.02 * n$
        obtain the corresponding *vertices* and *triangles* information from *cls*
        get the distance from the center of the car to the bottom of the car d_{bottom}
        for every vertice v in *vertices* **do**
            get the height of v, v_h
            if $v_h < -(v_{bottom} - d_{flood})$ **then**
                $v_h = -(v_{bottom} - d_{flood})$
        $n = n + 1$
        reproject 3D model with current *vertices* and *triangles* to 2D image using r and t from res_n, generate the mask,M_{curr}
        calculate the IoU of M_n and M_{curr}, IoU_{curr}
        if $IoU_{curr} < IoU_{last}$ **then**
            return water level d_{flood}

5 Experiments

In this chapter, the experiments involved in implementing the above method are conducted. The first section shows how different numbers of output classes affect the results. Once the number of output classes is determined, the model is further trained using ApolloCar 3D blurred dataset introduced in chapter 4. The second section will present the variation of the model's output keypoint error for different training iterations. In the third section, the model is further fine-tuned with ApolloCar 3D Flood dataset, and the evaluation metrics about bounding box will be illustrated.

The first three experiments obtain the optimal model parameters for Keypoint R-CNN. The learning rate for all training processes is 0.0025. Due to the GPU memory limitation, the batch size is set to 1. Next, the size of each car instance in the existing flood dataset is adapted. They are fed into the Keypoint R-CNN model separately to get the corresponding detection results. The optimal input size is obtained from the detection results.

The last set of experiments compares different PnP methods to determine the best pose estimation method.

5.1 Different Output Class Numbers

Keypoint R-CNN can obtain the class information of each car. Since the Apollo dataset gives 79 different types of cars, corresponding to 3 different classes of cars, i.e., SUV, Sedan, and Hatchback, the original Keypoint R-CNN pre-trained model will predict which of the 79 types each car belongs to, i.e., the class information. According to the introduction in the previous chapter, three models corresponding to SUV, Sedan, and Hatchback are picked out, respectively. Therefore, for the proposed method, it is only necessary to know which of these three classes a car belongs to, not which of these 79 types it is. This information is redundant for the proposed method. Therefore, the experiments are conducted for verifying whether the model output is more accurate with class number 3 or class number 79.

For this experiment step, ApolloCar 3D blurred dataset is used as the training set. This dataset aims to compare the accuracy of using 79 output classes and using three output classes at different resolutions. The classes of the annotated files of the new dataset are converted to the corresponding SUV, Sedan, and Hatchback classes. In other words, the 79 classes are converted into three classes. The existing model is fine-tuned using the new

class labels. As shown in Figure 5.1, after the images are input to the original model, 79 classes are output, and then these 79 classes need to be converted into 3 classes. When the original model is retrained, 3 classes will be output directly. Since the existing model is trained with 79 classes, the starting loss of class is large. To get a reasonable comparison result, the class accuracy of the model with three classes needs to be trained to the same class accuracy as the model with 79 classes. It is worth noting that both the 3 classes and the 79 classes are converted to 3 classes when evaluating the class accuracy. When the IoU of the ground truth's bounding box and the prediction's bounding box reached 0.5 or more, the two bounding boxes can be decided as a match. Assume that the number of all matching bounding boxes with the same class is N_{match} and the number of all matching bounding boxes is N_{all}, then the accuracy rate is as shown in Equation 5.1. Then the class accuracy of two different models is counted every 8000 iterations, as shown in Figure 5.2.

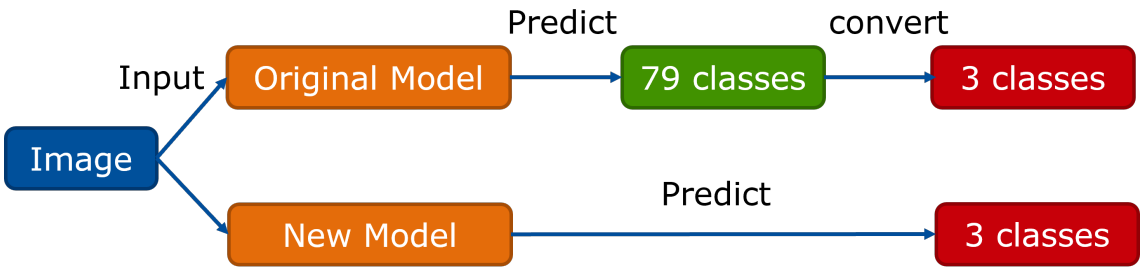
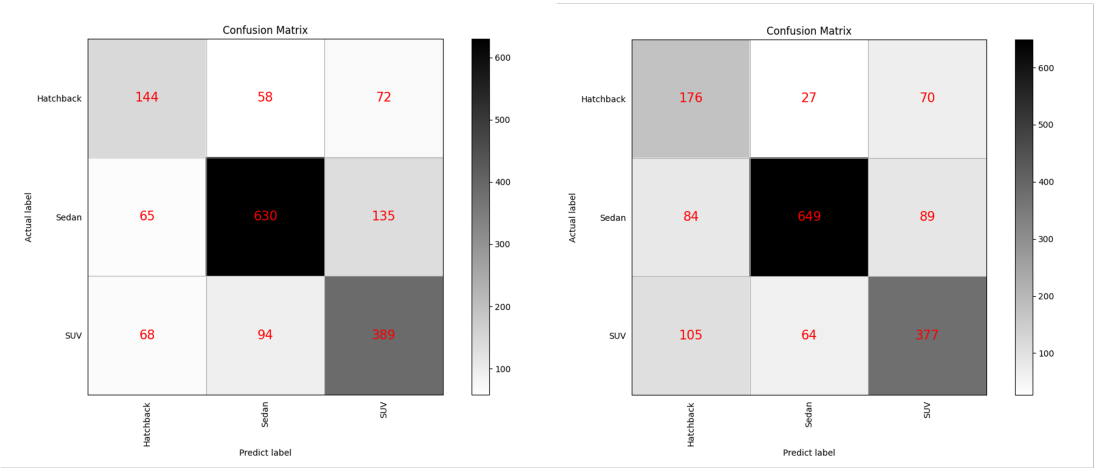


Figure 5.1: The difference between the original model and the new model

$$Accuracy = \frac{N_{match}}{N_{all}} \tag{5.1}$$

Figure 5.2: The accuracy of class with different class numbers

As can be seen from the figure, the number of output classes of the original model is 79, and each output class is converted into the corresponding three classes of Sedan, Hatchback,



(a) Confusion matrix of original model (b) Confusion matrix of new model

Figure 5.3: Confusion matrix of original model and new model

and SUV. The green line represents the accuracy of the converted classes. The newly trained model directly outputs classes of Sedan, Hatchback, and SUV, so that the class accuracy can be calculated directly using the output classes. The red line represents the classes accuracy of the new model. When the comparison starts, the accuracy of both are the same. In the subsequent training process, the accuracy of the three classes is higher overall than the accuracy of the 79 classes. This difference is mainly because the three classes are more generalizable as they can summarize the main features of these classes without distinguishing the differences between different types of models. From Figure 5.3, we can see the confusion matrix of the output classes of the two models. From the confusion matrix, the precision of each class for both models is obtained, as shown in Table 5.1. As can be seen from the table, except for the class precision of the SUV of the original model, which is slightly higher than that of the new model, the other two classes' precision of the new model is significantly higher than that of the original model. Otherwise, the class of Sedan has higher precision, while the hatchback class is more often incorrectly detected as SUV, which is the main reason for the low class precision results. Otherwise, the overall accuracy of the new model is better than that of the original model. The distribution of the classes in the ApolloCar 3D dataset can be seen in Figure 5.4. The hatchback class has fewer labels, while Sedan has the highest number of labels, which is one of the main reasons for the low class accuracy of the hatchback.

	Hatchback	**Sedan**	**SUV**	**Overall accuracy**
Class precision of original model	0.526	0.759	**0.706**	0.703
Class precision of new model	**0.645**	**0.790**	0.690	**0.732**

Table 5.1: Class precision of two models

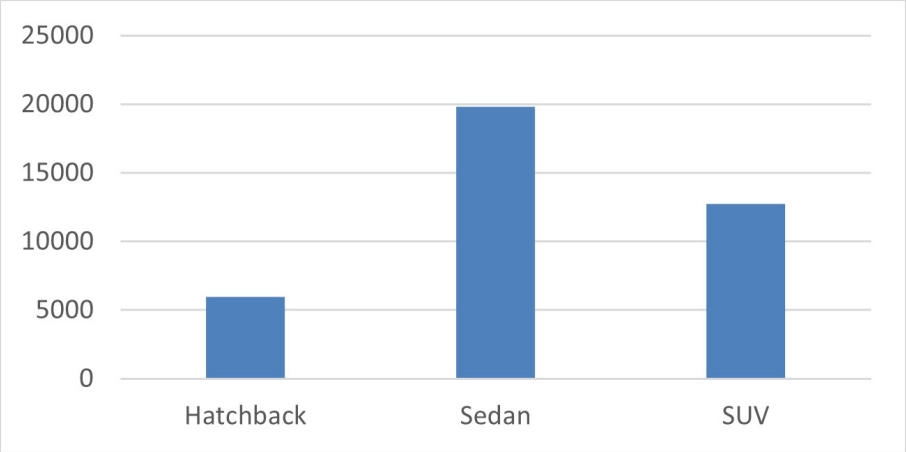


Figure 5.4: The distribution of the classes in the ApolloCar 3D dataset

After the experiment, it is shown through the data that the results with 3 classes are significantly better than the results with 79 classes. The output class number of 3 is final determined.

5.2 Keypoint Detection Results at Different Iterations

After determining the number of classes to be 3, the model continues to be trained. Because the output of the keypoints is tuned, keypoint evaluation metrics are made according to the method. When the IoU of the ground truth's bounding box and the prediction's bounding box reached 0.5 or more, the two bounding boxes are considered to be matched. There are 66 ground truth keypoints for each car. The keypoints are sorted by keypoint classes. Each keypoint has three pieces of information, x, y, and is_visible. Here, x and y represent the position of the key point on the image, and is_visible represents whether the key point is visible or not. The ground truth of each keypoint of the matched target is iterated, and if visible, calculate the pixel distance between this point and the corresponding predicted point. The distance error of all key points is summed up, and the average value is found. This mean value represents the final key point evaluation metric of the model. The equation is as follows:

$$E_{kpt} = \frac{\sum_{i=1}^{n} \sqrt{(x_i - \bar{x_i})^2 + (y_i - \bar{y_i})^2}}{n}. \tag{5.2}$$

The position of predicted keypoint on the image are presented as x_i and y_i. $\bar{x_i}$ and $\bar{y_i}$ present the position of ground truth keypoint. And n indicates the total number of keypoints used for evaluation.

In the method presented in the previous chapter, these detected keypoints are filtered out before using them. Therefore, there is no need to consider the accuracy of all detected key-

points. Also, the confidence threshold of the output keypoints is challenging to determine, so this keypoint evaluation metric is developed to be applicable to the method.

The same test set as in the previous section is used in order to test the variation in the results of keypoint error for different numbers of training iterations. As shown in Figure 5.5, this is the curve of keypoint error with the number of training iterations. As the number of iterations increases, the keypoint error tends to reach plateau. Since the pre-trained model of GSNet is used, the initial number of iterations is 130,000. After 6,000 iterations, the total training interations of the current model are reached to 190,000. The error decreases less. In order to prevent the overfitting brought by too many training rounds, when 200,000 iterations are reached, the training is ended. The keypoint error of the original model is 8.463 pixels, and after retraining, the keypoint error is reduced to 8.037 pixels. This indicates that the key point detection is more accurate after retraining.

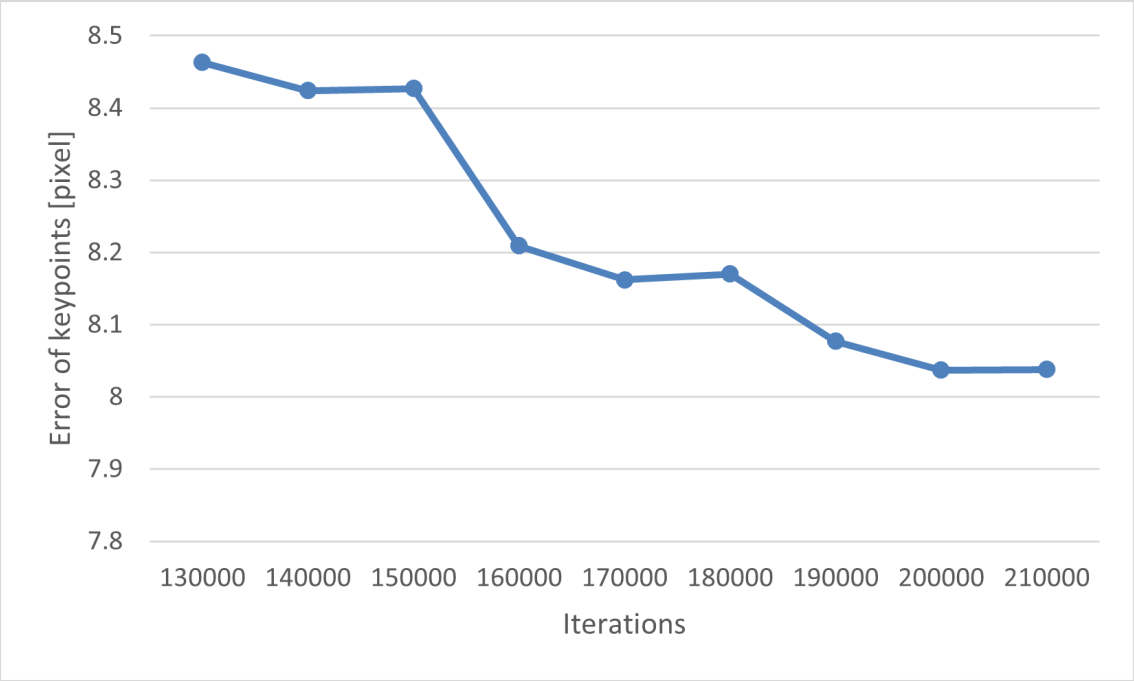


Figure 5.5: Keypoint error at different number of iterationst

5.3 Evaluation of Bounding Boxes

For this experiment step, the ApolloCar 3D Flood dataset is used as the training set. The purpose of using this dataset is to improve the detection rate of the model for submerged cars.

When the model that detects the optimal keypoints is determined, the parameter weights of the key point aspects are not updated in the next training. This is done because, in this dataset, each vehicle will be partially missing. However, this missing keypoint will

still be kept in the annotation file. If the keypoints are involved in the training, it will prevent the model from extracting the correct keypoint features.

Table 5.2 shows the different number of training iterations on the results. *Iteration* represents the number of iterations of fine-tuning on the previously trained model. As can be seen from the table, in the first 1000 iterations, *Recall* increases significantly, which means that the number of missed detections decreases significantly, and when it reaches 4000 iterations, the change of *Recall* tends to be stable. On top of that, it is not difficult to find that *precision* decreases as the number of training iterations grow. This means that after training, the model produces more false detections. However, for our method, the false detections can be filtered out by the results of Mask R-CNN. So in this experiment, we are more interested in the ability of the model to detect the target. The result is optimal when the number of iterations is 4000.

Iteration	**TP**	**FP**	**FN**	**Recall**	**Precision**
0	1171	**817**	548	0.681	**0.589**
1000	1667	2867	57	0.967	0.368
2000	1674	2296	50	0.971	0.422
3000	1661	2063	63	0.963	0.446
4000	**1686**	2395	**38**	**0.978**	0.413
5000	1660	1699	64	0.963	0.494
6000	1678	2125	46	0.973	0.441
7000	1680	2119	44	0.974	0.442
8000	1684	1993	40	0.977	0.458

Table 5.2: Bounding box evaluation matrix with different iterations

5.4 Instances of different sizes

As mentioned in the previous chapter, after the image is input to the Mask R-CNN model, each instance is extracted and placed in a black background image with a resolution of 3384*2710 after resizing. In the following experiments, an optimal size of the width of the car bounding box will be determined.

From Table 5.3, we can see that TP increases continuously when the width of the bounding box is not greater than 600, which indicates an improvement in the detection performance of the model. When the width of the bounding box is larger than 600, TP becomes smaller and smaller. This also confirms that the width of the bounding box has an effect on the detection of the model, and it is necessary to choose the appropriate instance size. Also, it can be seen from the table that *Precision* gets smaller as the width gets larger, which indicates that the model will have more output of errors as the instances are enlarged.

Width [pixel]	TP	FP	FN	Recall	Precision
200	100	**1**	66	0.602	**0.99**
300	116	4	50	0.699	0.967
400	119	21	47	0.717	0.85
500	125	45	41	0.753	0.735
600	**128**	40	**38**	**0.771**	0.762
700	123	60	43	0.741	0.672
800	124	82	42	0.747	0.602
900	127	87	49	0.705	0.574
1000	114	104	52	0.687	0.523
1100	111	159	55	0.669	0.411
1200	101	208	65	0.608	0.327
1300	95	177	71	0.572	0.349
1400	90	234	76	0.542	0.278
1500	82	242	84	0.493	0.253

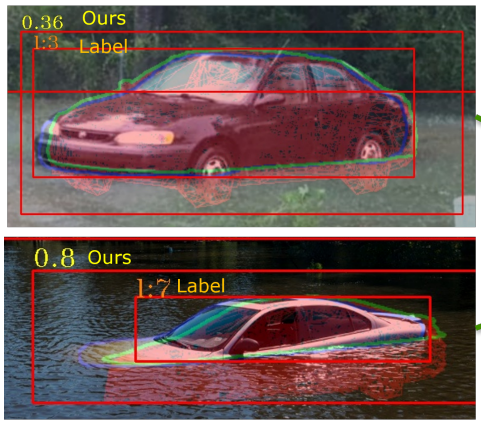
Table 5.3: Bounding box evaluation metrics with different size of instance width

5.5 Comparison of different PnP methods

In this section, 100 images with submerged cars from the ETHZ dataset are used to build a test set. In the introduction of Chapter 2, we know that the ETHZ dataset is divided into 11 levels. Each level has a fixed range of water depth. Since the ETHZ dataset does not provide centimeter-level water depth annotationand by comparing the actual observation with the label of ETHZ, there are some values that are not reasonable. As can be seen in Figure 5.6, the height of the water level exceeds the wheel's radius. Level 3 corresponds to a range of 10 cm to 21 cm, while the actual radius of the wheel is about 32 cm. Also, by actual measurement, the water level in the second image is around 80cm, which is also no longer in its level range. So these 100 images are annotated manually to get more accurate test results. As shown in Figure 5.7, the 3D model closest to the car in the images is displayed by software CloudCompare. The location of the submerged car on the 3D model is then determined and thus annotate the water depth. It is worth mentioning that a true value cannot be obtained because it is not possible to measure the flooded cars in the field. The error still exists through the proposed labeling method.

The results are output using different PnP methods. The OpenCV function-cv2.solvePnP() is used for the following experiments. OpenCV provides several solvers for PnP. Among them, EPNP and ITERATIVE are applied to the proposed method.

Table 5.5 and Figure 5.9 show the output using both methods, EPNP and ITERATIVE. Since the ITERATIVE method has a minimum input of 6 keypoints, the results cannot be output when the number of keypoints is 4. The average difference is used to evaluate the results of the water level estimation. The average difference of water level D is defined as shown in Equation 5.3. d_g is the ground truth of the manually labeled water level. d_p



Level Name	Range	Value nearest integer
	cm	cm
level0	No water	0.0
level1	0.0 - 1.0	1.0
level2	1.0 - 10.0	10.0
level3	10.0 - 21.25	21.0
level4	21.25 - 42.5	43.0
level5	42.5 - 63.75	64.0
level6	63.75 - 85	85.0
level7	85.0 - 106.25	106.0
level8	106.25 - 127.5	128.0
level9	127.5 - 148.75	149.0
level10	148.75 - 170.0	170.0

Table 1. Level to centimeters relation table

Radius:
~12.5inch = 31.75cm

Figure 5.6: Comparison of ETHZ labels with actual values

Figure 5.7: Annotation the water depth (example images under CC BY-NC-SA 2.0)

is the predicted water level. The average of the absolute values of d_g and d_p obtained from all matched cars is calculated. Tabel 5.5 show the change in the water level average difference as the number of key points increases.

$$D = \sum_{i=1}^{n}(|d_p - d_g|) \tag{5.3}$$

Keypoint number	4	6	8	10	12	14
EPNP [m]	0.169	0.162	0.155	0.152	**0.137**	0.145
ITERATIVE [m]	null	0.178	0.148	0.146	**0.136**	0.14

Table 5.4: Average difference of different keypoint number

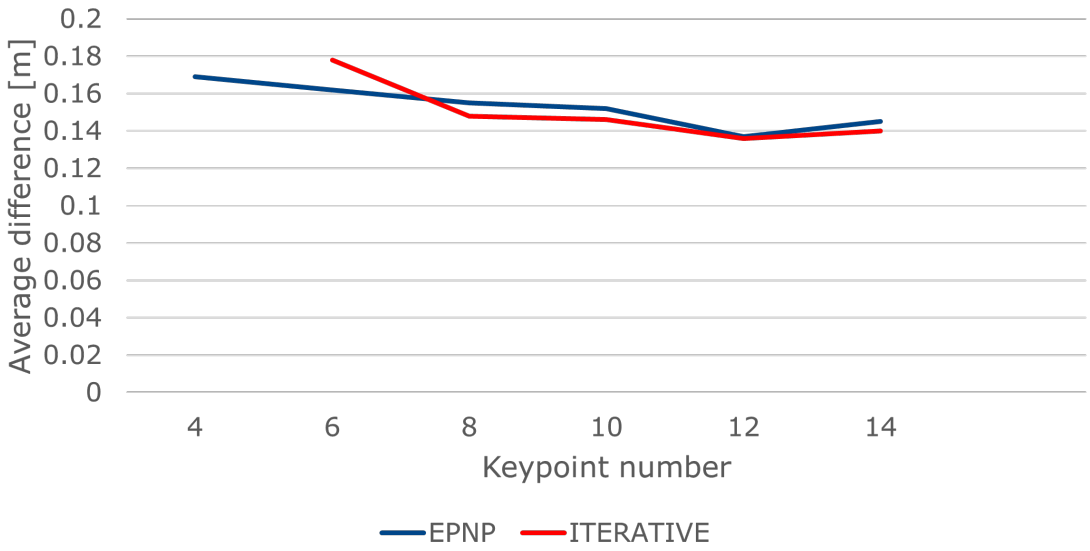


Figure 5.8: Average difference of different keypoint number

From the output, it can be seen that the average error decreases all the time when the number of key points is not greater than 12. The average difference increases when key points are greater than 12. Comparing the two methods, EPNP performs better when the number of keypoint is not greater than 6. When the keypoint is greater than 6, the method of ITERATIVE is consistently better than EPNP.

Keypoint number	4	6	8	10	12	14
EPNP	105	101	94	85	78	73
ITERATIVE	null	103	92	86	80	75

Table 5.5: Matched object number of different keypoint number

Table 5.5 and Figure 5.9 represent the relationship between the number of key points and the number of matched objects. Matched object represents the IoU of the mask generated by 3D model reprojection with Mask R-CNN generated by the car is greater than the

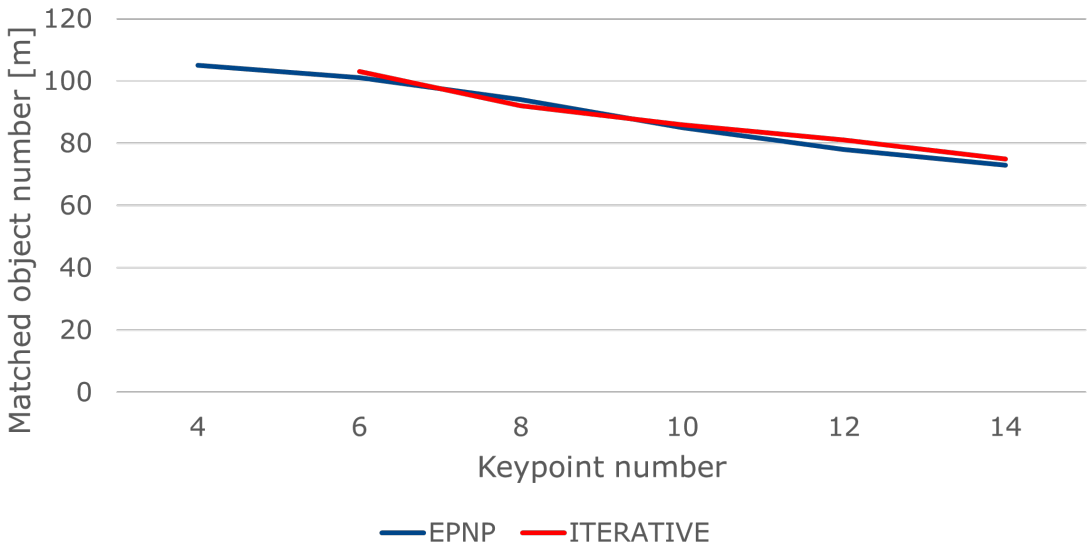


Figure 5.9: Matched object number of different keypoint number

set threshold. Here it is set to 0.5. When it is below the threshold value, the output is invalid. It can be seen from the output that as the number of key points increases, the number of matched objects decreases. The number of matched objects is similar for both methods compared. From Table 5.5, it can be seen that when the number of keypoints is 12, the average difference is the smallest. Therefore, the proposed method's number of key points is set to 12. When the number of keypoints is determined, it can be seen from the data that ITERATIVE is better than EPNP. In the method proposed in this thesis, ITERATIVE is used as the PnP solver.

Iterations	100	200	300	400
Average Difference	0.144	0.136	0.136	0.137

Table 5.6: Average difference of different iterations

When computing the pose information using the PnP solver, the keypoints are selected randomly by iteration and the best keypoints are chosen. Next, the optimal number of iterations is determined experimentally. As shown in Table 5.6, the average difference tends to stop changing when the number of iterations is greater than 200. It means that the optimal keypoints can be selected when the number of iterations is 200.

The method proposed in this thesis for pose estimation approximates RANSAC. An API for solving PnP pose estimation with RANSAC is provided in OpenCV. Therefore, the results output from it are used for comparison with the proposed method. With the same test set, there are 85 matched objects with an average difference of 0.172. The results show that the method optimized by us setting the number of iterations is better than RANSAC method.

6 Results and Discussion

6.1 Results

The experiments in Chapter 5 eventually lead to the following results.

- It is experimentally demonstrated that the accuracy of the model output is higher when the number of output classes is 3 compared to the conversion from 79 output classes to 3 classes.
- The model is retrained using the ApolloCar 3D Flurred dataset, and the keypoints of the model output are improved with different resolution test images.
- After fine-tuning the model using a dataset of synthetic floods, the ApolloCar 3D Flood dataset, the model's ability to detect cars in floods is improved.
- In order to make the fake camera intrinsic parameters closer to the real situation, the input images are the same size as the images in the Apollo dataset, and each car instance is resized to the suitable size. It is concluded from the experiments that the model can accurately detect the maximum number of objects when the width of each instance is 600 pixel.
- The results obtained by the EPNP and ITERATIVE methods are similar, and the results of the ITERATIVE method are slightly better than those of the EPNP method when the number of input keypoints is 12. After 200 iterations for each instance, the optimal pose information can be selected.
- The average difference in water level is 13.6 cm based on 80 detected cars on 100 images. The model achieved an accuracy of decimeter level.

Figure 6.1 shows the final visualization of each input image. The figure shows that the red mesh boundary lines represent the 3D model fitted to the 2D image. There are two color lines on the outer boundary of the car, where the blue line represents the edge of the car extracted by Mask R-CNN, and the red line represents the outer boundary of the 3D model projected to the 2D image. It can be seen from the figure that the lower edges of the two lines are very close to each other, which represents that the water surface of the flood determined by the 3D model after iteration is similar to the water surface determined by Mask R-CNN. The output water level, car class information, and confidence are in the upper left corner of the bounding box.

Figure 6.1: Visualization result of the proposed method

As can be seen in Figure 6.2, the proposed method performs well on all perspectives. This indicates that this method has good generalizability. It can be applied to pictures from multiple sources, such as the web, mobile phones, outdoor surveillance cameras, etc. It also indirectly proves that accurate projections can be obtained using fake camera intrinsic parameters.

Figure 6.2: Visualization result on different perspectives

However, many issues can cause incorrect results. Some results with large deviations between predicted values and ground truth are analysed and the causes are summarized as follows.

- Wrong keypoint match, resulting in reprojection failure, as shown in Figure 6.3.
- The occlusions caused by other objects, as shown in Figure 6.4.

- The car in the picture differs from the specific 3D model, as shown in Figure 6.5.

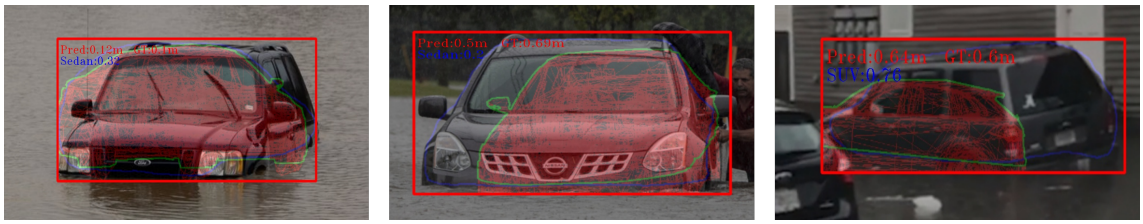


Figure 6.3: Wrong keypoint match

Figure 6.4: The occlusions caused by other objects

Figure 6.5: The large difference between the car on the picture and the 3D model

According to the ground truth of the water level, it is divided into five ranges, and the average difference of the water level within each range is calculated separately. As shown in Table 6.1, the best estimation results are obtained when the water level is between 0.4m and 0.6m. When the water level is too high or too low, it will make the detection result worse. If the water level is too high, the keypoint of the exposed water part is reduced, and the estimation of the pose can only be made by the keypoint of the exposed water with the 3D key point, which makes the result inaccurate. If the water level is too low, the splash of the car is too large, which will affect the detection results of Mask R-CNN, thus leading to deviation from the real results during the iterative estimation of the water level. The histogram of the water level average difference is shown in Figure 6.6. As can be seen from the histogram, the difference range for most of the data is between 0 and 0.2m, which indicates that the output is very accurate. However, there are still a small number of results with a large difference, which leads to a larger average difference in the test.

Water level Range [m]	**[0, 0.2)**	**[0.2, 0.4)**	**[0.4, 0.6)**	**[0.6, 0.8)**	**[0.8, inf)**
Average Difference [m]	0.247	0.157	0.102	0.111	0.134
Number of the cars	9	19	30	15	7

Table 6.1: Average difference of different water level range

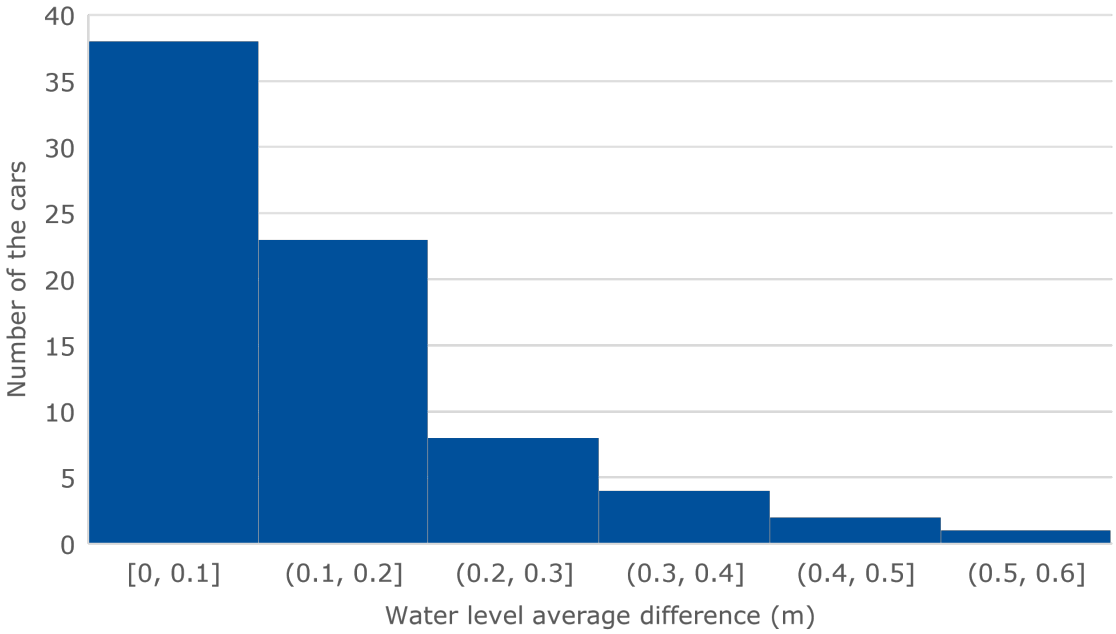


Figure 6.6: The histogram of the water level average difference

6.2 Discussion

Based on the three causes of errors presented above, the following analysis can be done.

The wrong keypoint matching comes from the error of keypoint detection on the one hand, the wrong output class on the other. If the error of detected keypoints is large or the number of detected keypoints is small, it is difficult to find the optimal keypoints randomly by iteration. At the same time, if the class is wrong, fitting with the wrong 3D model will lead to the incorrect estimation of pose information. The bottleneck in this part is the ability of the model to identify classes. In this thesis, different training sets are generated by various data enhancement methods to improve the model's ability. After retraining of the model, the performance of the model is improved. However, there is still a gap between the current simulated dataset and the real flood scenario, and there is no flood scenario that can be used for training. It is challenging to ensure the robustness of the model.

Since the method proposed in this thesis relies on detecting the part of the car that is flooded, if the car is obscured by other objects, such as cars, pedestrians or trees, it can lead to incorrect water level estimation results. Therefore, when selecting a picture as input, it should be ensured that the bottom of the car is not obscured.

The third cause of the error is that the vehicle in the image differs significantly from the 3 representative models. If the vehicle in the picture is too long, or the keypoint location is too different from the model's keypoint location, the pose information cannot be estimated correctly, and thus the correct water level estimation result cannot be obtained. The key to solve this problem is to label as many keypoints of the different 3D model as possible, while allowing the Keypoint R-CNN to recognize more classes of cars. This solution can potentially solve this problem. Due to the limited human resources in this thesis, it is not possible to label too many keypoints.

Various experiments on the selection of PnP solvers are made in this thesis. Notably, 12 random keypoints are used for the calculation to achieve the highest accuracy. The number of keypoints can be chosen according to the actual situation in the process. For example, to increase the number of detected objects, the number of key points can be reduced appropriately. If the number of detected keypoints in the image is small, it is also possible to use the EPNP method with 4 keypoints for the calculation.

7 Summary and Outlook

7.1 Summary

In this thesis, an approach based on a combination of deep learning and traditional computer vision is proposed to solve the problem of flood water level estimation. This is a multi-stage approach, first extracting the instances of each image by Mask R-CNN. Then the RGB images and masks of the instances are resized. After that, the resized RGB images are fed into Keypoint R-CNN to detect keypoints. The 2D keypoints and the corresponding 3D keypoints are solved using the PnP solver for the pose information. Water level estimation is performed by comparing the reprojected mask with the mask output from Mask R-CNN.

GSNet's model is used as a pre-trained model. The ApolloCar 3D dataset is image augmented, and new datasets, ApolloCar 3D Flood and ApolloCar 3D Blurred, are generated. Retraining the pre-trained model with these datasets enabled the model to improve the detection of submerged cars in flood scenarios.

In addition to this, to validate the method's performance, the ETHZ dataset is used for testing. 100 images with submerged cars in ETHZ are annotated manually. The average difference of the final water level is 13.6 cm, which achieves the estimation accuracy of the decimeter level.

The proposed method in this thesis has the following advantages.

- It can be applied to images from different cameras or sources. The fake camera intrinsic parameters are used for the pose estimation, which does not give the factual pose information but has accurate reprojection.
- After retraining, it can be applied to images of different resolutions.
- It is more flexible, generalizable, and accurate than other methods for flood level estimation.
- It is a 3D model-based water level estimation method, so it is more reasonable than the methods that use the pixel distance on the image to estimate the water level.
- The method can detect the class of cars to perform a more accurate fit during reprojection.

However, there are some drawbacks to the method. For example, its performance can be affected by the quality of the keypoints. In addition, it is not an end-to-end model and requires the output from two neural network models and the traditional computer vision method for water level estimation.

7.2 Outlook

There are a lot of promising future work and interesting exploration directions to improve our research.

First, it is a multi-stage approach. Therefore, it is difficult to apply to real-time water level estimation. A promising direction is to convert the multi-stage to an end-to-end method and compress the model reasonably to improve the computational speed.

Second, due to the lack of a large number of annotated flood datasets, the training can only be done with simulated flood datasets. The flood dataset should be built specifically to improve the model effect in the following work.

Finally, because only three models are labeled with key points, these three models cannot fit all models well, resulting in the imperfect fitting. Therefore, as many models with significant differences as possible should be labeled in the following work to get a better fitting effect.

Bibliography

[1] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Advances in neural information processing systems 28 (2015) 91–99.

[2] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, IEEE Transactions on Pattern Analysis Machine Intelligence (2017).

[3] T. Y. Lin, M. Maire, S. Belongie, J. Hays, C. L. Zitnick, Microsoft coco: Common objects in context, Springer International Publishing (2014).

[4] X. Song, P. Wang, D. Zhou, R. Zhu, C. Guan, Y. Dai, H. Su, H. Li, R. Yang, Apollocar3d: A large 3d car instance understanding benchmark for autonomous driving, in: IEEE, 2018.

[5] P. Chaudhary, S. D'Aronco, M. Moy de Vitry, J. P. Leitão, J. D. Wegner, Flood-water level estimation from social media images, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 4 (2/W5) (2019) 5–12.

[6] B. Tellman, J. Sullivan, C. Kuhn, A. Kettner, C. Doyle, G. Brakenridge, T. Erickson, D. Slayback, Satellite imaging reveals increased proportion of population exposed to floods, Nature 596 (7870) (2021) 80–86.

[7] C. Sazara, Methods for detecting floodwater on roadways from ground level images (2021).

[8] L. Deng, D. Yu, Deep learning: methods and applications, Foundations and trends in signal processing 7 (3–4) (2014) 197–387.

[9] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[10] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[11] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[12] X. X. Lu, A review of solutions for perspective-n-point problem in camera pose estimation, in: Journal of Physics: Conference Series, Vol. 1087, IOP Publishing, 2018, p. 052009.

[13] R. Hartley, A. Zisserman, Multiple view geometry in computer vision (cambridge university, 2003), C1 C3 2 (2013).

[14] B. Zhou, Z. Chen, Q. Liu, An efficient solution to the perspective-n-point problem for camera with unknown focal length, IEEE Access 8 (2020) 162838–162846.

[15] G. Bradski, A. Kaehler, Opencv, Dr. Dobb's journal of software tools 3 (2000) 2.

[16] S. Roweis, Levenberg-marquardt optimization, Notes, University Of Toronto (1996).

[17] V. Lepetit, F. Moreno-Noguer, P. Fua, Epnp: An accurate o (n) solution to the pnp problem, International journal of computer vision 81 (2) (2009) 155–166.

[18] J. H. Challis, A procedure for determining rigid body transformation parameters, Journal of biomechanics 28 (6) (1995) 733–737.

[19] S. V. Stehman, Selecting and interpreting measures of thematic classification accuracy, Remote sensing of Environment 62 (1) (1997) 77–89.

[20] D. M. Powers, Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation, arXiv preprint arXiv:2010.16061 (2020).

[21] M. Geetha, M. Manoj, A. Sarika, M. Mohan, S. N. Rao, Detection and estimation of the extent of flood from crowd sourced images, in: 2017 International Conference on Communication and Signal Processing (ICCSP), IEEE, 2017, pp. 0603–0608.

[22] Z. Meng, B. Peng, Q. Huang, Flood depth estimation from web images, in: Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Advances on Resilient and Intelligent Cities, 2019, pp. 37–40.

[23] Y. Feng, C. Brenner, M. Sester, Flood severity mapping from volunteered geographic information by interpreting water level from images containing people: A case study of hurricane harvey, ISPRS Journal of Photogrammetry and Remote Sensing 169 (2020) 301–319.

[24] S. Park, F. Baek, J. Sohn, H. Kim, Computer vision–based estimation of flood depth in flooded-vehicle images, Journal of Computing in Civil Engineering 35 (2) (2021) 04020072.

[25] L. Wang, S. Guo, W. Huang, Y. Qiao, Places205-vggnet models for scene recognition, arXiv preprint arXiv:1508.01667 (2015).

[26] B. A. Kharazi, A. H. Behzadan, Flood depth mapping in street photos with image processing and deep neural networks, Computers, Environment and Urban Systems 88 (2021) 101628.

[27] L. Ke, S. Li, Y. Sun, Y.-W. Tai, C.-K. Tang, Gsnet: Joint vehicle pose and shape reconstruction with geometrical and scene-aware supervision, in: European Conference on Computer Vision, Springer, 2020, pp. 515–532.

[28] K. G. Derpanis, Overview of the ransac algorithm, Image Rochester NY 4 (1) (2010) 2–3.