UNIVERSITY OF TWENTE

MASTER'S THESIS

Pose Estimation with Deep Neural Networks Trained on Tiny Datasets

Author:
Yueying LIANG

Supervisor:
Dr. Michael YANG

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

INRIA
EEMCS

January 20, 2022

Declaration of Authorship

I, Yueying LIANG, declare that this thesis titled, **?**Pose Estimation with Deep Neural Networks Trained on Tiny Datasets**?** and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UNIVERSITY OF TWENTE

Abstract

Michael Yang, Christian Duriez
EEMCS

Master of Science

Pose Estimation with Deep Neural Networks Trained on Tiny Datasets

by Yueying LIANG

This report presents my project on the pose estimation from training a deep neural network with small datasets. The primary object of this project comes from the project on a soft medical robot. Since soft robots have unlimited degrees of freedom and convenience, being hard to control is a great challenge. A camera and a sensor are attached to it to have enough information to control the robot's end-effector. The camera can shoot videos while the robot is working, and the sensor provides the pose information of the end-effector. However, the implementation of the sensor is not only complicated but also expensive. Fortunately, the camera pose collected by the sensor can now be estimated with deep learning algorithms.

Deep neural networks usually perform much better on large datasets. What comes with the great advantage of the performance of deep neural networks is the constraints on the dataset size. A pruning strategy on the most state-of-the-art deep learning frameworks is proposed to overcome the limited performance while training with small datasets, which is more common in practical use. According to experimental results, translation and rotation test errors are reduced after training a model on small datasets. Therefore, the strategy could also be promoted to other applications when the training dataset is of limited size.

Acknowledgements

Thanks to my first-year study at the University of Twente, I am armed with solid knowledge and fully prepared for the robotics and computer vision field project. I'm grateful for my education at UT and every professor who guided me. I could never make it without them.

Thanks to my supervisor Michael Yang. He has been a great support during my study; he is always there and offers help whenever I am in need. He gives valuable advice when I'm confused and unsure about the next step. He is like the lighthouse, always there and guides my way.

Thanks to my supervisor Christian Duriez and other colleagues. They offered me great help when I did the project with them. Because of the pandemic, I couldn't do experiments at the lab; my colleagues were there to support me, offered whatever materials I needed, and arranged tutorials to help me get familiar with their work.

Thanks to my friends and parents, they have been the strongest support during my study and encouraged me when I got lost. I'm more than grateful for their care about me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Compared to traditional rigid robots, soft robots naturally have countless advantages due to their flexibility. Similarly, as sonar technologies, soft robots are also inspired by nature. With the term **soft**, the mechanics of the robots usually rely on deformable structures. The materials used to build a soft robot are much more compliant than rigid robots, thus significantly improving the safety of the robots since soft robots could avoid many possible collisions that rigid robots might have with the environment while working. Moreover, combined with the redundant actuation, a soft robot's infinite degrees of freedom will make it possible for complicated manipulations in constrained space. Such benefits have made soft robots competitive in many fields, especially for medical and surgical study, for soft robots have much less damage to bodies and unlimited operabilities while treating patients.[1]

However, what comes along with the great benefits is the great difficulty of simulating and controlling such a robot. Although the difficulties are exactly the causes of the slow progress in studying soft robots, plenty of work has been done to overcome the difficulties. Previous methods to simulate soft robots include voxel-based discretization[2] and Finite-Element-Method(FEM)[3],[4].

INRIA is one of the teams that has been investigating the soft robots, and this project is done within the team led by Doctor Christian Duriez. The INRIA team proposed a FEM-based simulation tool called SOFA years ago, and the simulation tool has become one of the most widely used tools worldwide.[5] SOFA is an open-source framework that focuses on real-time simulation; it is especially emphasized in the field of medicine and surgeons. From all possible advantages, SOFA has conspicuous ones over other tools of the same category in:

- Flexible:
  - Efficient design and prototyping of simulation
  - Multiple representations
  - Multi-physics
- Modular with its plugin-activation system

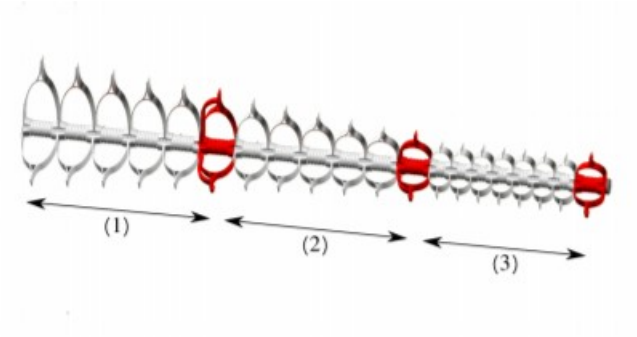

FIGURE 1.1: Main Structure of the robot[6]

- Interactive
  - User interactions to take into account
- Efficient:
  - Multi-threading
  - GPU computing

INRIA has developed a deformable manipulator robot based on a compliant spine with all these advantages[6]. Spine animals inspire it. Each section of a spine can move in a certain region. By attaching a couple of sections, the deformable robot can move in a larger working space and accomplish tasks in a rather narrow space. The robot is designed and controlled with a method based on numerical models and simulations. It is modeled by FEM and controlled by a closed-loop control strategy. Simulation and real-world experiments have shown the advantages of the robot, including the large workspace it could work in; dexterity so that the end-effector of the robot could move in more than one direction in the task space; ability to avoid obstacles while moving, and compliance to the environment so that it will not cause damage to the patient.

The structure of the robot is shown in Fig 1.1 and a picture of the robot is shown in Fig 1.2

This project is based on this deformable robot. Although it has shown great effectiveness, obstacles remain that hinder the robot from being widely used. Unacceptable costs and the complexity of settling are part of those. For example, apart from the camera attached to the end-effector to help with the robot's vision, a sensor is also attached to collect the pose information of the end-effector. However, suppose the pose information of the robot could be estimated from the images and videos shot by the monocular camera attached. In that case, the implementation of the sensor can be safely removed, thus could considerably cut the budget and simplify the settling process.

This project will be focusing on the pose estimation of a camera according to the images and videos the camera obtains. It is a computer vision task, and obviously, a great amount of work has been

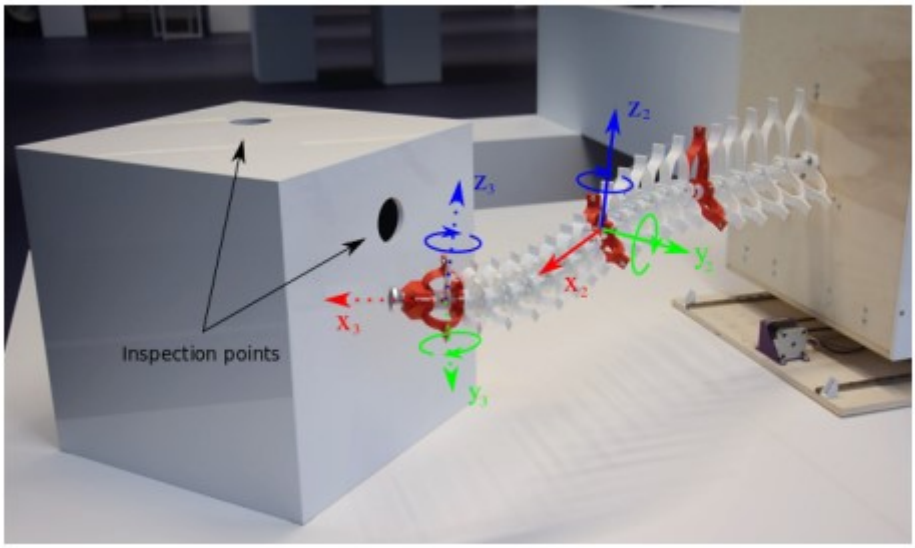

FIGURE 1.2: Real photo of the robot[6]

done in the task, and with much prior knowledge of a camera, the pose information can be easily calculated years ago. Nevertheless, in more recent work, researchers have been bringing deep learning to this task; with the algorithm, the pose information could be predicted even without prior knowledge of a camera.

The most recent work is called **Unsupervised Scale-consistent Depth and Ego-motion Learning from Monocular Video**.[7] It was published in 2020 with an end-to-end framework. This paper has been a success in estimating the depth of an image and the pose of a camera since the experiments on KITTI and NYU datasets show improvements in previous work. However, recent work is based on an extremely large dataset that is not usually available for practical use. When a dataset is small, a neural network with great expression ability can cause an overfitting problem. Fortunately, model pruning is an efficient method to deal with the problem. Moreover, since pruned model is sparse, the inference process could also be accelerated, which will be a bonus while in practical use by saving time and computation costs.

Above all, the primary objectives of this project are:

- Improving the accuracy on deep neural network models while training with small datasets.
- Accelerating the inference process.

The neural network and model pruning algorithm will be further introduced in the following sections. The structure of this project is:

- Related Work: In this section, the evolution of the related study will be presented. Comparisons among different methods will also be presented.

- Methodology: In this section, the design of the robot, the most recent deep learning neural network, and model pruning algorithm will be introduced.
- Experiments and Discussion: In this section, experiments on the KITTI dataset will be listed, including dataset preparation, experiments setup, and evaluation of their performance.
- Conclusion: In this section, the conclusion will be drawn.

Chapter 2

Related Work

2.1 Visual Odometry

For the last thirty years, great effort has been put into developing an accurate and robust monocular Visual Odometry(VO) system. The workflow of the conventional method is illustrated in Fig 3.1. As shown in the figure, the pipeline is consisted of calibrating the camera, detecting features, matching or tracking features, rejecting outliers, estimating motions and scales, local optimizing, etc. Moreover, it has been treated as a golden rule for good performance. However, a classic algorithm usually has many constraints; for example, specific design and fine-tuning are required for different tasks to ensure performance. Moreover, prior knowledge of the camera is also essential to estimate the absolute scale.

2.1.1 Supervised Learning

DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks[8] is the first research that introduces deep learning into visual odometry, it proposes an end-to-end deep learning method to skip camera calibration while estimating the pose of a camera for the first time. It is a network-based on Recurrent Convolutional Neural Networks(RCNNs). Thanks to its end-to-end training manner, it could estimate the poses of a camera directly from a sequence of raw RGB images. Moreover, DeepVO can learn effective feature representation from CNN and model sequential dynamics and relations between frames with the help of deep RNN.

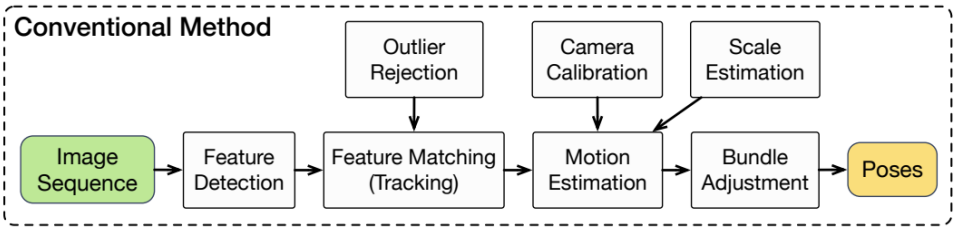

FIGURE 2.1: Workflow of Conventional VO system[8]

DeepVO training is a supervised learning process; although it has its drawbacks, it brings new insights to researchers.

2.1.2 Unsupervised Learning

Although supervised learning methods have shown effectiveness in single view depth estimation and visual odometry, labeled data is not always available in practical use, especially for the enormous dataset. Therefore, several self-supervised learning models appeared in the following three years.

Unsupervised Learning of Depth and Ego-Motion from Video[9] is the first one of them. The self-supervised process is achieved by training the depth and pose estimation networks simultaneously. The task of view synthesis is used as the supervisory signal. Therefore, the networks could be coupled during training, yet separated during testing.

UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning[10] is another work that focuses on self-supervised learning networks. Apart from the advantage of self-supervision, UnDeepVO also achieves another salient feature – absolute scale recovery. To do so, the UnDeepVO is trained by using stereo image pairs. Uniquely, monocular images are used to test, so the network is considered a monocular system. However, it is not acceptable when stereo image pairs are not available.

Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction[11] is another unsupervised learning algorithm. Similar to UnDeepVO, this paper also adopts stereo sequences to learn depth and visual odometry. Spatial and temporal warp errors are allowed using stereo sequences since stereo image pairs can correct spatial errors, and relationships between frames could correct temporal warp errors. Moreover, the framework could estimate the depth of a single image and two-view odometry from a monocular image sequence during the test.

Digging Into Self-Supervised Monocular Depth Estimation[12] proposes a sequence of improvements on previous unsupervised learning methods. The paper proposes a minimum re-projection loss to handle possible occlusions more robustly. Moreover, the paper also proposes a novel full-resolution multi-scale sampling method so that the visual artifacts could be greatly reduced.

GANVO: Unsupervised Deep Monocular Visual Odometry and Depth Estimation with Generative Adversarial Networks[13] introduces a generative unsupervised learning algorithm. With deep convolutional Generative Adversarial Networks(GANs), the proposed framework could estimate the 6-DOF pose information of a camera and monocular depth map of the scene from raw RGB image sequences.

Unsupervised Scale-consistent Depth and Ego-motion Learning from Monocular Video[7] stresses the challenges that remain in the previous unsupervised learning frameworks, including limitations of the performance caused by unknown moving objects, the existence of these objects violate the underlying static scene assumption in geometric image reconstruction. The paper proposes a kind of geometry consistency loss for scale-consistent predictions to overcome the challenges. In addition, the paper also proposes an induced self-discovered mask to deal with unrecognized moving objects and occlusions.

D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry[14] is one of the most state-of-art frameworks focusing on Monocular Visual Odometry. The novel D3VO framework exploits deep neural networks on three levels for monocular visual odometry– deep depth, deep pose, and deep uncertainty. The paper first proposes a self-supervised monocular depth estimation network trained on stereo videos. Particularly, D3VO separates the stereo image pairs with predictive brightness transformation parameters. Thus they could have similar lighting conditions. Additionally, the paper also models the photometric uncertainties on the input image pixels to further improve the depth estimation performance.

2.2 Model Pruning

However, unsupervised learning models normally take advantage of large size of datasets. Therefore, the accuracy will suffer when applying it on a small, unlabeled dataset. When the model is too large, maybe too wide or too deep for a dataset, the model will also learn about outliers and noises. The extra expression ability will make inference accuracy drop. Model pruning is helpful for this situation; by cutting some weights off, the expression ability of a model reduces to match the size of a dataset. Tons of previous works have attempted to apply pruning strategy to the models.

The pruning strategy is now widely used in computer vision and classification tasks. No pruning has been attempted to apply on visual odometry tasks so far. All kinds of pruning strategies can be roughly divided into three categories: Unstructured Pruning, Structured Pruning, and Pattern Pruning.

2.2.1 Unstructured Pruning

The unstructured pruning, i.e., irregular pruning, was first proposed in[15]. In this paper, models are compressed irregularly with an iterative and heuristic method. Although the compression rate is limited, it opened the gate of pruning deep networks and offered a new thought to accelerate the training and inference process and make deep neural networks dealing with small datasets possible.

The work[15] is further improved by[16] and[17] with a more powerful optimization framework – The alternating direction method of multipliers (ADMM)[18]. With the ADMM optimization framework, the compression ratio of unstructured pruning greatly increases while retaining a comparable accuracy.

However, designing and successfully implementing a powerful model requires an efficient algorithm and hardware. As for the pruning strategies, the coordinates of the pruned weights need to be stored in RAM. Unfortunately, the overheads for unstructured pruning are especially high because of the irregularity in the coordinates of the unstructured pruned weights. Therefore, structured pruning started to come into researchers' sights.

2.2.2 Structured Pruning

On the contrary of unstructured pruning, structured pruning was first proposed by[19], and[20] to deal with the unacceptable overheads on the hardware. Thus there is regularity in the pruning strategy. More specifically, structured pruning usually refers to channel pruning and filter pruning, channel pruning prunes columns, and filter pruning prunes rows.

The main benefit of structure pruning is saving storage on indices, for the pruned weights exist in a row or column. Structured pruning could dismiss all the unimportant weights in a model irregularly; the compression ratio is considerable. Nevertheless, on the contrary of the unstructured pruning, to guarantee a comparable accuracy, structured pruning sacrifices the compression ratio to benefit on overheads.

2.2.3 Pattern Pruning

To better handle the trade-off between unacceptably high overheads and low compression ratio, pattern pruning became the first attempt in[21]. Theoretically, pattern pruning has some regularities in patterns, thus reducing some overheads compared to unstructured pruning, and since patterns could differ in different channels, the compression ratio could be increased compared to structured pruning. Unfortunately, the accuracy in[21] is too low to be widely used.

Later, by applying the pruned filters to DNNs before pooling, another work[22] made pattern pruning more popular.

The pruning strategies will be further elaborated in Chapter 3.

Chapter 3

Methodology

This chapter introduces one of the most state-of-art monocular visual odometry framework – SC-Depth[23], and the improvements on it. Compared to D3VO, SC-Depth has a significant advantage apart from performance, SC-Depth is an end-to-end framework, which simplifies the training process a lot. Moreover, different from most datasets that are used to train visual odometry models, like KITTI datasets and Cityscapes, dataset from INRIA is completely collected from a monocular camera. Fortunately, compared to TrianFlow[24], SC-Depth framework does not require stereo image pairs to train on visual odometry tasks. Thus, the SC-Depth model would be the perfect model to work on.

But these frameworks all have a problem, which is the training is based on large datasets. Therefore, when large datasets are not available, overfitting will be a serious problem that greatly infects the performance of the model. Fortunately, model pruning has been an effective strategy to deal with the problem. The pruned model will solve the overfitting problem and save computation and time costs by accelerating the training and inference process.

3.1 SC-Depth Model

Previous supervised learning frameworks usually take advantage of labeled data, however, it is not always available to label data in practical use. Luckily, unsupervised learning models help solve the problem.

As the novelty stated in the paper, SC-Depth Model takes advantage of consecutive frames sampled from unlabeled videos to train a model to learn depth. Afterwards, with the knowledge of predicted depth and relative pose between two frames, the *Photometric Loss* will supervise the model to learn about pose of the camera. Additionally, the *Geometry Consistency Loss* and the *Self-Discovered Mask* are introduced to deal with the problems of scale inconsistency and moving objects, respectively.

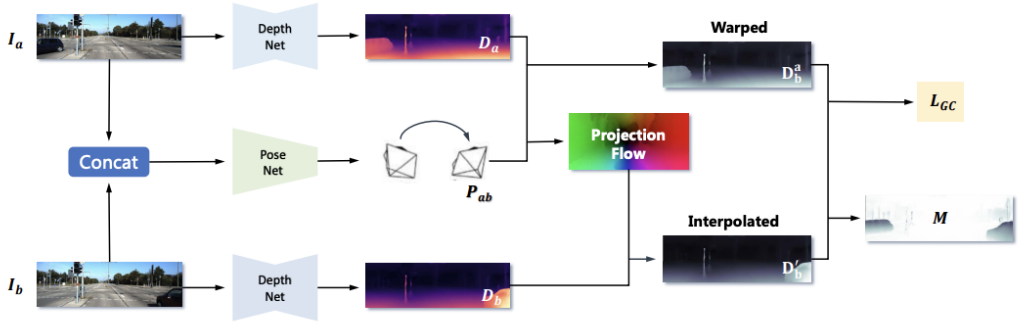

FIGURE 3.1: Workflow of the SC-Depth model[23]

3.1.1 Overview

The overview of the workflow is illustrated in Fig 3.1.[23]

As indicated in the figure, I_a and I_b are two consecutive frames sampled from a video without labels, the images are firstly fed into Depth Networks to predict depth maps (D_a, D_b).

At the same time, the 6-DOF pose of the camera P_{ab}

is estimated from the relationship between the two frames (I_a, I_b) with the Pose Network. With the knowledge of predicted depth maps and predicted pose of the camera, reference image I'_a could be generated by interpolating input image I_b.[25][9]. Then, the network could be trained under the supervision of the Photometric Loss between the reference image I'_a and the input image I_b.

Nevertheless, scale inconsistency remains a problem due to the movements of camera and objects. Similarly, as the objects moving in the scene, obscure and occlusions could happen as well. Both problems have negative influences on the performance of the model. Here states the motivation of authors to introduce the Geometry Consistency Loss and Self-Discovered Mask strategy. One thing is worth noting that the Photometric Loss is weighted by generated mask in the final loss function.

Finally, the total loss of the framework is a combination of the scaled three losses, i.e., the weighted Photometric Loss, the Geometry Consistency Loss, and a smoothness Loss which is adopted directly from previous work.

3.1.2 Loss Functions

Photometric Loss

Similarly as in[25][9], with the difference between synthesized image I'_a and reference image I_a, the Photometric Loss function is defined as Equation 3.1

$$L_P = \frac{1}{|V|} \sum_{p \in V} \left\| (I_{a(p)} - I'_{a(p)}) \right\|_1 \tag{3.1}$$

V in the equation represents the set of successfully projected dots from the image plane I_a to the image plane I_b. And the $|V|$ stands for the number of successfully projected points in set V. L1 norm is used here as the loss because of its high robustness to outliers. However, according to[26], to deal with the illumination changes in real-world applications, an additional image dissimilarity loss *SSIM* is required to be added to the photometric loss function. Thus, the Photometric Loss is modified to Equation 3.7

$$L_P = \frac{1}{|V|}\sum_{p\in V}(\lambda_i \left\|(I_{a(p)} - I'_{a(p)})\right\|_1 + \lambda_s \frac{1 - SSIM_{aa'(p)}}{2}) \tag{3.2}$$

Smoothness Loss

The Smoothness Loss is adopted from[27] to regularize the predicted depth maps since the photometric loss itself is not sensitive to textures in real-world scenarios. Same as[27], the smoothness function is Equation 3.3

$$L_s = \sum_p (e^{-\nabla I_a(p)} \cdot \nabla D_a(p))^2 \tag{3.3}$$

Geometric Consistency Loss

To guarantee the geometric consistency while training, the depth map of one image should also be consistent with the warped depth map of the other image by the predicted 6-DOF pose information. Thus, the geometric inconsistency could be reduced by minimizing the difference between predicted depth maps of the two mentioned depth maps. Because the last image from a batch is still consecutive to the first image in the next batch, this consistency will pass on not only in the same batch but also in the whole sequence. For each $p \in V$, the mathematic expression of the difference $D_{diff}(p)$ is shown in Equation 3.4.

$$D_{diff}(p) = \frac{\left\|(D^a_b(p) - D'_b(p))\right\|}{D^a_b(p) + D'_b(p)} \tag{3.4}$$

The depth map $D^a_b(p)$ is obtained by warping the predicted depth map D_a with P_{ab}. By the reason that the warping flow does not lie on pixel grids, the predicted depth map D_b can not be directly used here to calculate the difference as analyzed before. Alternatively, to have the depth map D_b aligned with the warped depth map $D^a_b(p)$, D'_b is used and obtained by interpolating the depth map D_b. Moreover, the difference is normalized by dividing the sum of the two depth maps. The advantage of doing so is that normalized differences will equally treat points on different depth levels. Besides, normalized loss could contribute in statistical stability in training as well.

To consider the point gaps as a whole, the Geometric Consistency Loss could be defined as Equation 3.5

$$L_{GC} = \frac{1}{|V|} \sum_{p \in V} D_{diff}(p) \tag{3.5}$$

By minimizing the distance between the two depth maps, geometric consistency is guaranteed and passed on to the whole sequence frame by frame.

Self-Discovered Mask

Object moving and occlusions could be a disaster while training a model on multiple tasks, depth prediction models suffer most since the depth of objects that are moving or occluded by other objects can not be estimated reliably anymore. Previous work has tried a couple of solutions to deal with the problem. For example, introducing additional optical flow[28] and introducing semantic segmentation network[29] are some of the options. The methods are effective but require too much additional computation costs.

Luckily, with the definition and knowledge of D_{diff}, this problem could be solved much more easily by generating a self-discovered mask with the difference between depth maps. It is quite an intuitive idea, because ideally the difference between the two depth maps are supposed to be the same. If the gap is too large, it must have been a result from moving objects or occlusions. In addition, as described earlier, since the difference D_{diff} is normalized, this mask could be simply defined as Equation 3.6 to illustrate the opposite of the difference D_{diff}.

$$M = 1 - D_{diff} \tag{3.6}$$

So the mask could assign weights for the inconsistent pixels, i.e., small M indicates possible disturbances like occlusions and moving objects. The impact of the frames facing this inconsistency could be mitigated by adding the weighted mask M in the Photometric Loss. The modified loss function is shown in Equation 3.7.

$$L_p^M = \frac{1}{|V|} \sum_{p \in V} (M(p) \cdot L_p(p)) \tag{3.7}$$

Total Loss

Now that all the three losses are defined and modified, each work for a different purpose. The total loss will be a combination of the scaled three losses. The total loss function is given in Equation 3.8.

$$L = \alpha L_p^M + \beta L_s + \gamma L_{GC} \tag{3.8}$$

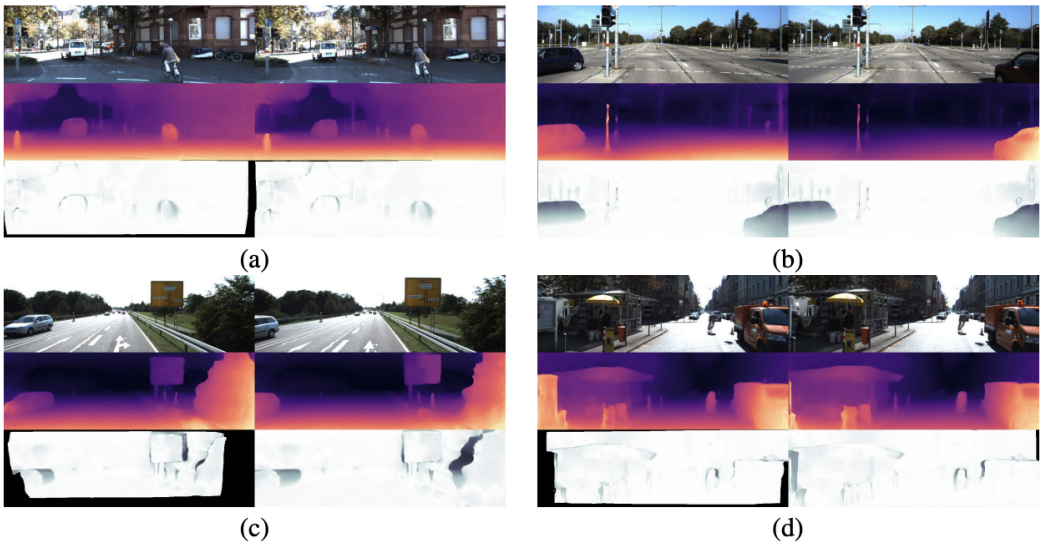

FIGURE 3.2: Visualization Results of Mask Map[23]

The escalators α, β, γ will be hyperparameters given by operators. The visual mask map is shown in Fig 3.2. The figures are raw images, depth maps and self-discovered masks from top to bottom, respectively. In the self-discovered mask figures, the darker part are more likely to be moving objects or occlusions. The lighter part are still scenes. The visual results are especially convincing in *(b)*, the two vehicles are the moving objects in the sampled images. And they are accurately detected in the self-discovered mask map.

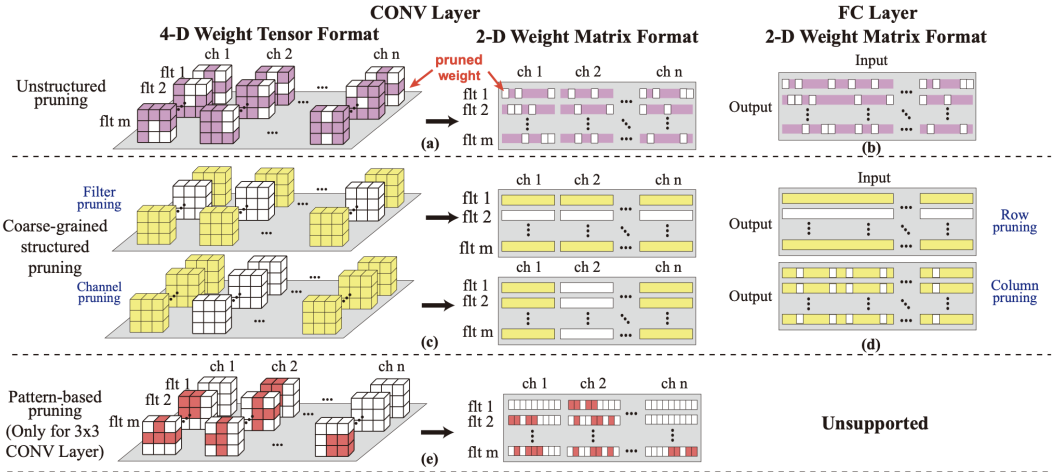

FIGURE 3.3: Scheme of Structure Pruning[30]

3.2 Model Pruning

As mentioned in the related work section, many previous works have been focusing on model pruning. Under a computer vision-related context, convolution layers and fully connected layers Pruning strategies can be classified as structure pruning, pattern pruning and irregular pruning. The pruning strategies will be elaborated in the following subsections. The scheme of different pruning strategies is illustrated in Fig 3.3.

3.2.1 Irregular Pruning

Irregular pruning, as the name indicates, removes weights at arbitrary positions. As indicated in the Fig 3.3 *(a)* and *(b)*, the 4D dimensional weight matrices of the convolution layers are first reshaped in 2D weight matrix format. Then the weights in the convolution layers and the fully connected layers are pruned to 0 in an arbitrary manner. Finally, the convolution layers will be reshaped back in 4D dimension format. The white boxes in the figure indicate pruned weights yet the colored boxes represent the remaining weights.

There is a trade-off between compression rate and accuracy. Normally, the accuracy will drop as the compression rate increases since the expression ability of the model is impaired; the balance between compression rate and accuracy should be taken extra care of while pruning. In other words, pruning should be done while making the accuracy drop acceptable as a premise. The accuracy of the pruned model should be comparable to the original model. When the premise is satisfied, one could consider increasing the compression rate; otherwise, the pruning will make no sense.

The compression rate of irregular pruning will be the most exiting in all pruning strategies since it will learn and cut the least important weights in a model. Irregular pruning also has its own disadvantages

though, for example, in the hardware content, enormous number of the irregular indices of the pruned weights are required to be stored, thus increases the overhead and influences the whole performance.

3.2.2 Structured Pruning

Structured Pruning usually refers to filter pruning and channel pruning. The pruning scheme is shown in Fig 3.3 *(c)*, *(d)*.

The 2D convolution and fully connected weight matrices are defined as filter pruning by pruning the whole row and channel pruning by pruning the whole column. Similarly as irregular pruning, while applying structure pruning on convolution layers of a model, the first step is to reshape the weight matrix in a 2D shape format. Then the pruned 2D convolution weight matrix is reshaped back to the shape of the original weight matrix.

However, compared to irregular pruning, the drawbacks of structured pruning are relatively limited compression rate. The main reason is that the channel and filter pruning strategies will cause greater accuracy drop than irregular pruning. But since the indices of pruned weights are regular, storing them in memory could save some time and energy, thus accelerates the training and inference process.

3.2.3 Pattern Pruning

Pattern Pruning is more like a compromise between the irregular pruning strategy and the structure pruning. Because usually there are some regularities in a pattern, such as pruning a whole column or a whole row in each filter of a kernel. Additionally, patterns applied to different filters can be different, thus there could be some irregularities as well. Consequently, pattern pruning could balance the trade-off much better. Compared to structure pruning, pattern pruning could achieve higher compression ratio with a comparable accuracy; compared to irregular pruning, pattern pruning could save some storage on hardware. The scheme is shown in Fig 3.3 *(c)*.

Now that pattern pruning has all these advantages, the remain problem for this strategy is how to choose patterns for kernels. According to enormous experimental results on different models, the basic rule of applying pattern pruning is to keep the middle weight in the kernel from pruning. Possible patterns are shown in red blocks in the Fig 3.4.

Note that the gray filters shown in the Fig 3.4 are pruned weights in *Connectivity Pruning*[31], the essence of which is adding filter pruning in pattern pruning. Both pruning strategies can be implemented in the same algorithm. The motivation of implementing filter pruning in pattern pruning is that by selectively cutting some connections between certain input and output channels, the compression ratio and speed for training and inference will increase

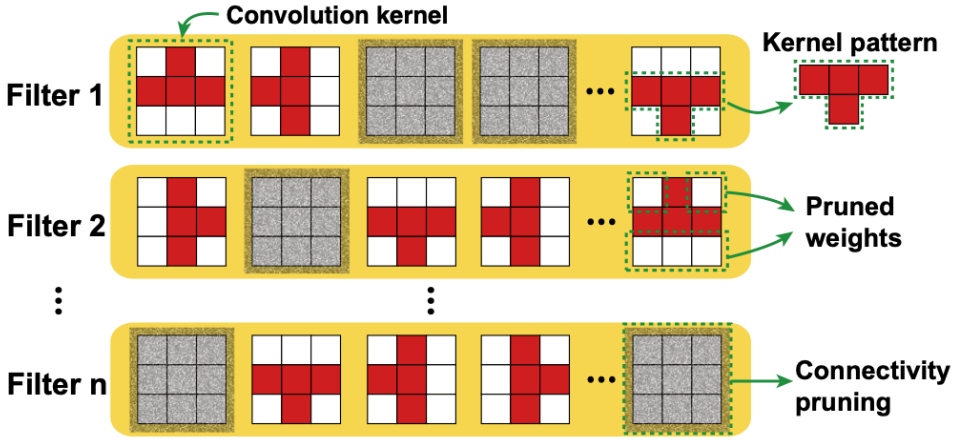

FIGURE 3.4: Illustration of pattern pruning and connectivity pruning[31]

even further[32]. Thus, connectivity pruning will be a great supplement to pattern pruning.

3.3 ADMM Regularized Optimizations

The alternating direction method of multipliers (*ADMM*)[18] is an algorithm that solves convex optimizations problems by decomposing them into two sub-problems, so that each of them could be solved separately and iteratively.[33]

Consider an DNN network has N layers and each layer is denoted as the *i-th* layer. Correspondingly, the weight matrices and bias matrices are denoted as W_i and b_i. Therefore, the loss function could be represented with:

$$\mathcal{L}\{W_i\}_{i=1}^{N}, \{b_i\}_{i=1}^{N}$$

Then the loss function of the model can be minimized with constraints subject to pruning strategy. The global problem can be rewritten as:

$$\begin{aligned} \underset{\{W_i\},\{b_i\}}{\text{minimize}} \quad & \mathcal{L}\{W_i\}_{i=1}^{N}, \{b_i\}_{i=1}^{N} \\ \text{subject to} \quad & W_i \in S_i, i = 1, ..., N \end{aligned} \tag{3.9}$$

S_i stands for the constraint sets of Pruning strategy.

Similar to ADMM-NN[34], the ADMM-Regularized Optimization Process is iterative. With the constraints in Function 3.9, the loss can not be minimized with the conventional stochastic gradient descent (SGD) method.[35] Fortunately, it can be solved with ADMM by decomposing the problem to two small problems and solve them iteratively.

As the detailed inference process presented in[36], the problem can be solved by several steps:

- Step 1: Reformulate Problem 3.9

$$\begin{aligned} \underset{\{W_i\},\{b_i\}}{\text{minimize}} \quad & \{\{W_i\}_{i=1}^{N}, \{b_i\}_{i=1}^{N} + \sum_{i=1}^{N} g_i(Z_i) \\ \text{subject to} \quad & W_i = Z_i, i = 1, ..., N \end{aligned} \tag{3.10}$$

- Step 2: Decompose the problem 3.10 into two sub-problems by applying Augmented Lagrangian Method[18]

$$\underset{\{W_i\},\{b_i\}}{\text{minimize}} \quad \{\{W_i\}_{i=1}^{N}, \{b_i\}_{i=1}^{N} + \sum_{i=1}^{N} \frac{\rho_i}{2} \left\| W_i - Z_i^t + U_i^t \right\|_F^2 \tag{3.11}$$

$$\underset{\{Z_i\}}{\text{minimize}} \quad \sum_{i=1}^{N} g_i(Z_i) + \sum_{i=1}^{N} \frac{\rho_i}{2} \left\| W_i^{t+1} - Z_i + U_i^t \right\|_F^2 \tag{3.12}$$

- Step 3: Solve the problem 3.11 with SGD and solve the problem 3.12 by:

$$Z_i^{t+1} = \Pi_{X_i}(W_i^{t+1} + U_i^t) \tag{3.13}$$

In the whole solving procedure, Z_i is an auxiliary variable; U_i stands for dual variable and t is the index of iteration. The hyperparameter ρ_i represents the scalar assigned to the L2 regularization as a penalty. Π_{X_i} is the Euclidean projection to $X_i \in \{S_i\}$. Additionally, to solve the problem iteratively, U_i is updated by Equation 3.14 until the convergence is guaranteed.

$$U_i^t = U_i^{t-1} + W_i^t - Z_i^t \tag{3.14}$$

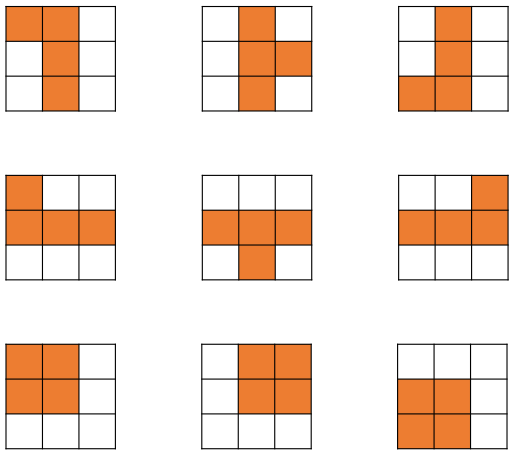

FIGURE 3.5: Example Patterns

3.4 Training Procedure

Now that pattern pruning has the highest performance overall as analyzed in previous section, it will be mainly implemented in this project. While applying pattern pruning strategy in training, there are also several steps to follow:

- Step 1: load a pre-trained model.

  To shorten the training process and achieve higher accuracy, a pre-trained model should be loaded, the higher accuracy, the better.

- Step 2: Initialize the ADMM parameters.

- Step 3: Choose Patterns.

  In this step, a set of patterns are given, containing different pattern designs. The patterns are binary matrices of the same shape with the weight matrices need to be pruned. Some example patterns are shown in Fig 3.5. The white blocks represent 0's and the colored blocks stand for 1's in the pattern matrix.

  The patterns are designed manually, note that according to practical experiments, the middle element in a weight matrix should be prevent from pruning. The number of patterns in a set is given as a hyperparameter, the best pattern for each kernel will then be chosen from the set based on the L2 norm.

- Step 4: Apply patterns on weight matrices.

Then the selected best pattern will be applied on the weight matrices by dot multiplying them together. Thus the weights in corresponding positions with 0's in the patterns are removed and the others remain.

- Step 5: Train the model.
- Step 6: Apply patterns on gradient weight matrices.

  Normally pruned weights will be updated after applying gradient weight matrices. To keep the weight matrices being hard-pruned, the same pattern must also be applied on corresponding gradient weight matrices. To do so, the pruned weight matrices are read, positions of 0's will be found in the weight matrices, and then the same mask composed of 0's and 1's is applied on gradient weight matrices, similarly as Step 4, element-wisely.

- Step 7: Update the ADMM parameters.

  Finally, the ADMM parameters are updated as introduced in the previous section at each training step.

Chapter 4

Experiments and Discussion

In this section, details of experiments are presented. Additionally, results, evaluation, and comparisons with previous work and discussions also come along.

4.1 Experiment Setup

4.1.1 Platform and Datasets

Platform

The experiments are implemented in the PyTorch framework. The project was run on a server with eight GPUs provided by the University of Texas at San Antonio; the models of the GPUs are Quadro RTX 6000.

Datasets

Similarly as in the paper[7], two datasets, Cityscapes[37] and KITTI[38] are used here to show effectiveness of model pruning and the improvements it could get while training on small datasets.

- Cityscapes:

  Cityscapes is a large-scale dataset shot in different cities in Europe. Cityscapes provide annotations for 30 classes grouped into eight categories and focus mainly on semantic segmentation tasks.

  Some example input images of Cityscapes are shown in Fig 4.1 and Fig 4.2.

- KITTI:

  Unlike Cityscapes, KITTI consists of a set of image sequences and their corresponding ground truths of camera poses. Moreover, the ground truths are the main superior over Cityscapes. Therefore, KITTI has been used to train and test performance on deep learning algorithms under visual odometry tasks in all kinds of previous work. Normally sequences 00-07 are used for training, 08 is used for evaluating,

FIGURE 4.1: Example image of Cityscapes[37]

FIGURE 4.2: Example image of Cityscapes[37]

FIGURE 4.3: Example image of KITTI[38]

FIGURE 4.4: Example image of KITTI[38]

FIGURE 4.5: Example image of INRIA Dataset

and sequences 09-10 are used for testing. Even in unsupervised deep learning algorithms, ground truths of the camera are still essential for testing.

Some example raw images of KITTI are shown in Fig 4.3 and Fig 4.4.

- INRIA Dataset: Apart from Cityscapes and KITTI datasets, which are public, there is also a private dataset provided by INRIA, and it was shot by the camera attached to the robot. It is not in real use under medical context though, the dataset contains two short videos shooting a QR code, and the scene changes slightly along with the movements of the camera. As mentioned above, the main problem of the dataset is too small and it brought my thought of improving the performance of training SC-Depth on small datasets is an interesting and meaningful task.

  Some example raw images of INRIA Dataset are shown in Fig 4.5 and Fig 4.6. Unfortunately, in this project, experiments on INRIA Dataset is not possible because the intrinsic parameters of the camera are unknown and not able to be collected because of pandemic.

FIGURE 4.6: Example image of INRIA Dataset

Reproduce the Results in the Paper

After 200 epochs of training, the rotation and translation error of sequence 09 in test set could be plotted as in Fig 4.7 and Fig 4.8, respectively. Similarly, the rotation and translation errors of sequence 10 are plotted in Fig 4.9 and Fig 4.10.

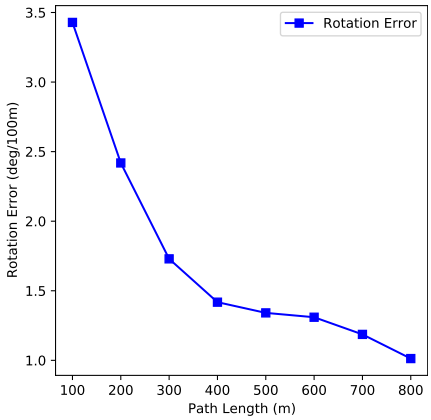

FIGURE 4.7: Rotation Error of Sequence 09

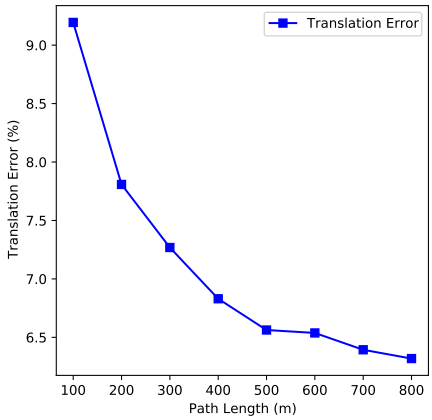

FIGURE 4.8: Translation Error of Sequence 09

Last but not least, the trajectory plot of the camera pose is also used to evaluate the accuracy of prediction. The trajectory plots

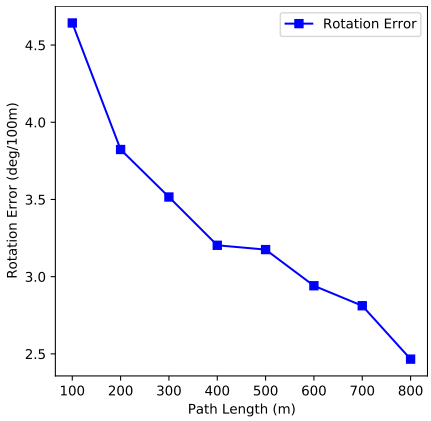

FIGURE 4.9: Rotation Error of Sequence 10

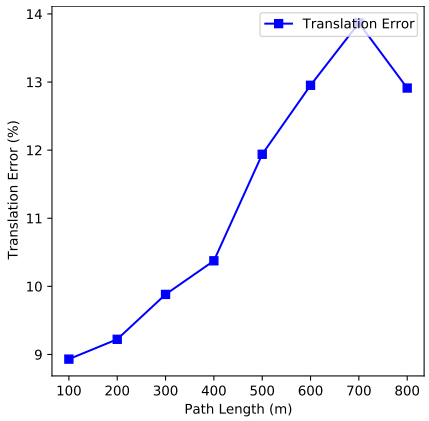

FIGURE 4.10: Translation Error of Sequence 10

for sequence 09 and 10 are shown in Fig 4.11 and Fig 4.12, respectively.

From the figures, we could tell that when the training set is relatively large, the prediction of the unsupervised training framework is reliable.

4.1.2 Preparation and Hyperparameters

Preparation

- Pretrain:

  Now that the real training set is small, a pre-trained model becomes beneficial. Cityscapes have similar features as real training sets and are thus suitable for pretraining. Additionally,[7] uses Cityscapes to pre-train the model as well; doing the same thing makes the performance of the experimental results more convincing. To make it compatible with the input size of the KITTI dataset, the raw images are resized to the same size as the KITTI dataset.

- Prepare the dataset:

  To prove the disadvantages of the proposed model on small datasets and the effectiveness of improvements on small datasets with model pruning strategy. The training set is reduced to $\frac{1}{7}$ of the original size. Therefore, only one sequence is used for training, and evaluation and test sets remain the same. The same training dataset is first used to train the original network and then used to train the pattern-pruned network. The same dataset will be used for testing to guarantee fairness in evaluation. To further strengthen the generalization of the pruning strategy, the experiments are extended by changing

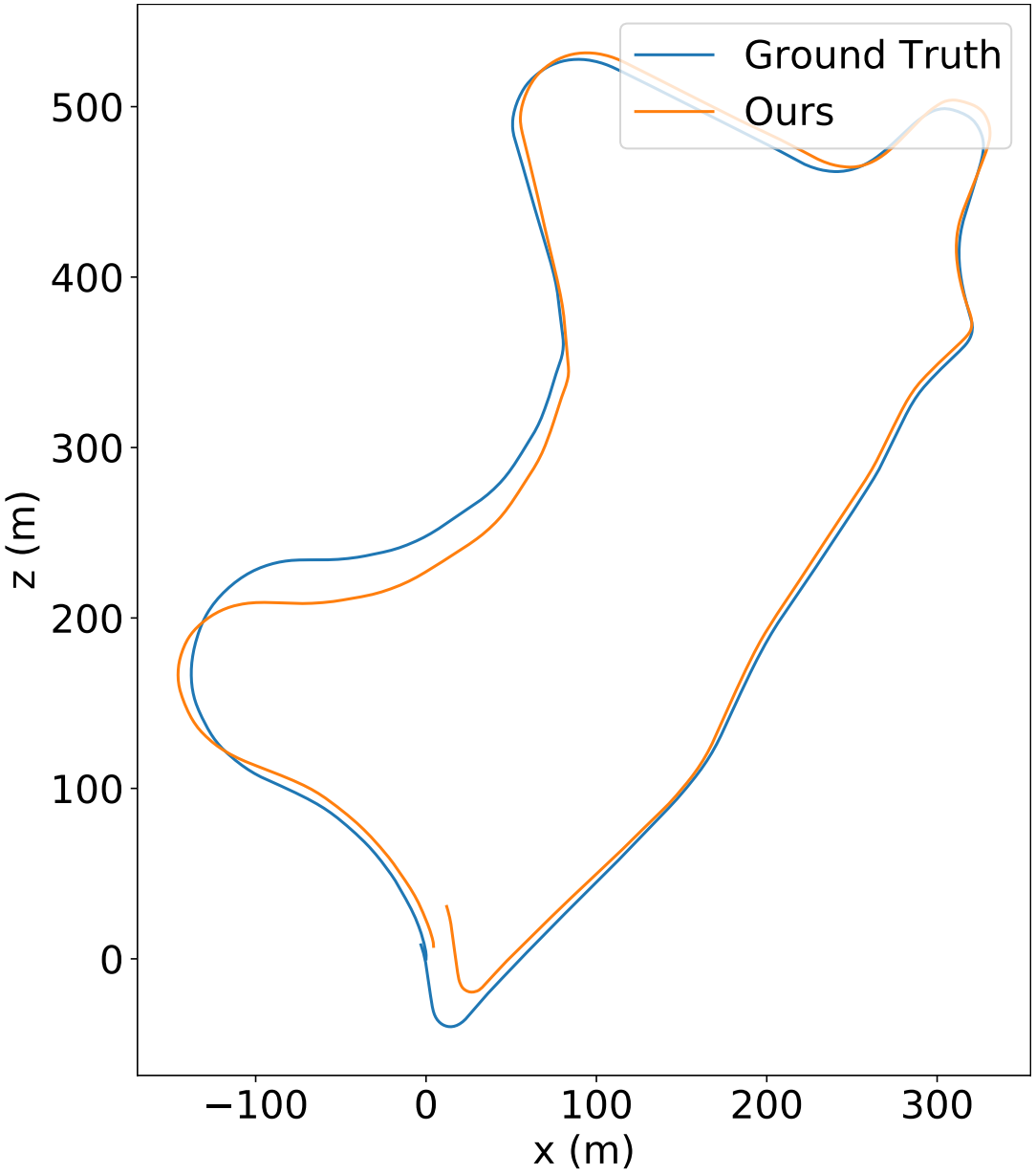

FIGURE 4.11: Trajectory Plot of prediction on sequence 09

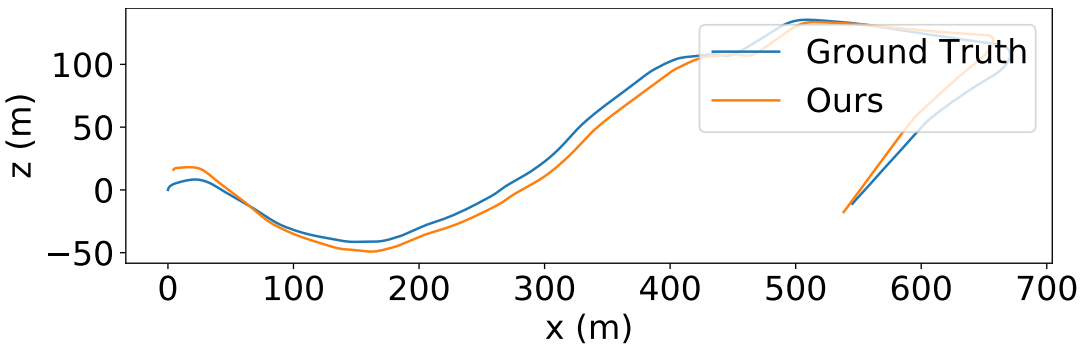

FIGURE 4.12: Trajectory Plot of prediction on sequence 10

training sets a couple of times, such as using sequence 00, 01, or 02 for training.

Hyperparameters

To ensure the consistency with[7], the hyperparameters are set to be the same:

- Batch Size: 4
- Learning Rate: 10^{-4}
- Scalars in Function 3.8: $\alpha = 1.0$, $\beta = 0.1$ and $\gamma = 0.5$
- Epochs: 200

Pruning Strategy

Unstructured pruning, structure pruning and pattern pruning are all used in this project to evaluate the efficiency of the pruning strategy while training a deep neural network with small datasets. In the structured pruning, filter pruning is applied as an example, and in the pattern pruning, the pattern set is the same as Fig 3.5. Only the convolution layers are pruned, and according to the pattern designs, the sparsity of the pruned layers is 55.6%. To ensure the fairness of comparison among pruning strategies, the sparsity ratio is set to be the same, i.e. 55.6%.

Model	Trainset Size	Sequence 09	
Zhou et al.[39]	8 Seq	Trans. Err(%) 17.84	Rot. Err(°/100m) 6.78
SC-Depth[7]	8 Seq	Trans. Err(%) 8.24	Rot. Err(°/100m) 2.19
SC-Depth	1 Seq	Trans. Err(%) 18.36	Rot. Err(°/100m) 5.83
Irregular Pruned SC-Depth	1 Seq	Trans. Err(%) 12.24	Rot. Err(°/100m) 4.29
Structure Pruned SC-Depth	1 Seq	Trans. Err(%) 17.05	Rot. Err(°/100m) 5.13
Pattern Pruned SC-Depth	1 Seq	Trans. Err(%) 13.00	Rot. Err(°/100m) 4.75

TABLE 4.1: Numerical comparison between models of KITTI sequence 09

Model	Trainset Size	Sequence 10	
Zhou et al.[39]	8 Seq	Trans. Err(%) 37.91	Rot. Err(°/100m) 17.78
SC-Depth[7]	8 Seq	Trans. Err(%) 12.45	Rot. Err(°/100m) 3.46
SC-Depth	1 Seq	Trans. Err(%) 21.08	Rot. Err(°/100m) 7.20
Irregular Pruned SC-Depth	1 Seq	Trans. Err(%) 15.59	Rot. Err(°/100m) 5.61
Structure Pruned SC-Depth	1 Seq	Trans. Err(%) 24.17	Rot. Err(°/100m) 9.72
Pattern Pruned SC-Depth	1 Seq	Trans. Err(%) 11.43	Rot. Err(°/100m) 4.99

TABLE 4.2: Numerical comparison between models of KITTI sequence 10

4.2 Results

In this section, numerical and visualization results are given to evaluate the improvements that pruning strategy could make while training on small datasets.

4.2.1 Error Comparison

Each sequence in sequence 00-07 is used once for training an original network and the pruned network. The mean translation error and rotation error are compared in Table 4.1 and Table 4.2.

Table 4.1 and Table 4.2 together indicate that when the dataset size is greatly reduced, the translation error and rotation error are

significantly increased. For instance, the translation error of sequence 09 trained by SC-Depth for the large dataset is 8.24% while the translation error trained by the same model for the small dataset is 14.92%, which is increased 1.8 $\times$. The rotation error of sequence 09 trained by SC-Depth for the large dataset is 2.19 °/100m, while the rotation error trained by SC-Depth for the small dataset is 5.116 °/100m, which is 2.33 $\times$ greater. Nevertheless, the test errors on sequence 10 increase much less. When training SC-Depth with large and small datasets, the translation error is comparable, 12.45% on large datasets and 12.89% on a small dataset.

However, although the test error of training on a small dataset increases a lot compared to the SC-Depth algorithm trained with a large dataset, the performance is still superior over certain previous work to some extent. For example, the translation error of sequence 09 trained by SC-Depth for the small dataset is 14.92%, yet the translation error trained by Zhou et al.[39] is 17.84%. Moreover, the rotation error of sequence 09 is also reduced from 6.78 °/100m to 5.12 °/100m. The reduction is even more for sequence 10 since the translation error of[39] on sequence 10 is 37.91%, and the translation error of SC-Depth trained on the small dataset is only 12.89%, which is a reduction of 1.94 $\times$. As for the rotation error, it is also reduced from 17.78 °/100m to 5.03 °/100m, which is a reduction of 2.53 $\times$.

When it comes to the performance of pruned SC-Depth model while training on small datasets, although different pruning strategies obtain different results, overall both translation error and rotation error on sequence 09 and sequence 10 reduce compared to normal SC-Depth. Take sequence 09 as an example; the translation error of irregular pruned, structure pruned and pattern pruned SC-Depth are 12.24%, 17.05% and 13.00%, respectively, all of them are smaller than normal SC-Depth trained on small datasets, i.e., 18.36%. Compared to previous work[39], the error reduction on sequence 10 is even more significant: translation error of pattern pruned SC-Depth is reduced by 2.32 $\times$, and the rotation error is reduced by 2.56 $\times$. Therefore, it indicates that the pruned SC-Depth is more powerful while dealing with small datasets.

Theoretically, irregular pruning surpasses the other two pruning strategies on sparsity since it could filter out the weights that make least significance on performance and prune them. However, the ideal sparsity for different tasks differs and it usually takes long time for experiments to find out. In this experiment, to compare the performance among different pruning strategies, the sparsity is set to be the same. As shown in the results, while testing on sequence 09, the error of irregular pruned SC-Depth is the smallest among the three pruning strategies. However, while testing on sequence 10, pattern pruned SC-Depth performs the best. Nevertheless, no matter which pruning strategy performs the best, experimental results obviously have shown that both irregular and pattern pruning strategy surpass

the basic SC-Depth. Thus, it is safe to conclude that involving pruning strategy can make improvements while training a deep neural network on small datasets.

4.2.2 Visualization Results

In this subsection, the visualization results of SC-Depth and Pruned SC-Depth trained on small datasets will be presented, the predicted and true trajectory paths are given for the baseline (base SC-Depth) and pruned SC-Depth by different pruning strategies.

From the visualization plots, the accuracy of prediction can be seen more intuitively. Similar conclusion could be drawn from this subsection as well: pruned SC-Depth could predict the trajectory more accurately.

Basic SC-Depth

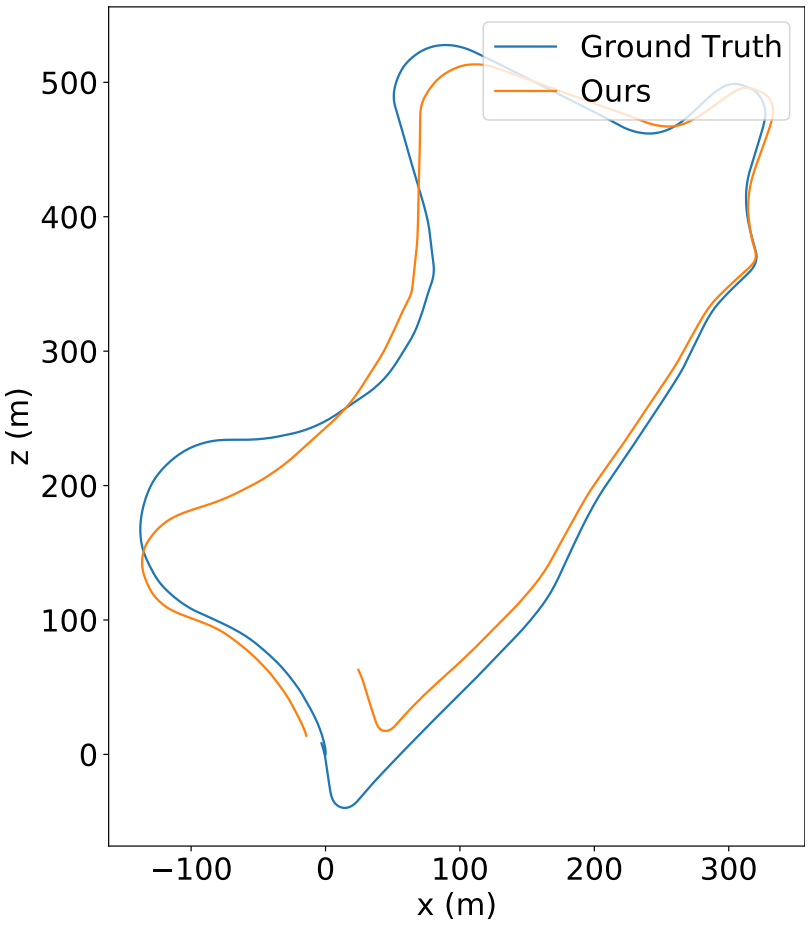

FIGURE 4.13: Trajectory Plot on sequence 09

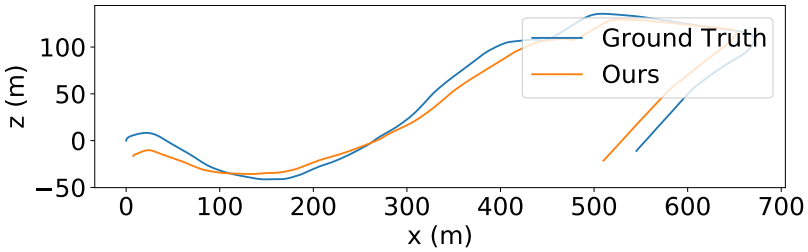

FIGURE 4.14: Trajectory Plot on sequence 10

Irregular Pruned SC-Depth

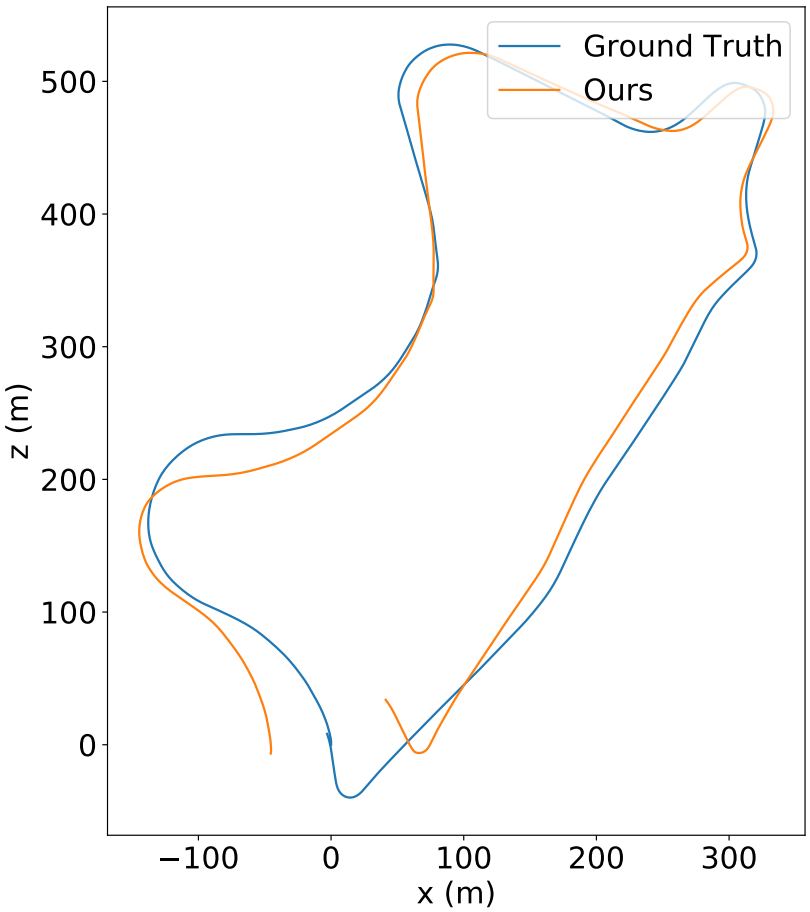

FIGURE 4.15: Trajectory Plot on sequence 09

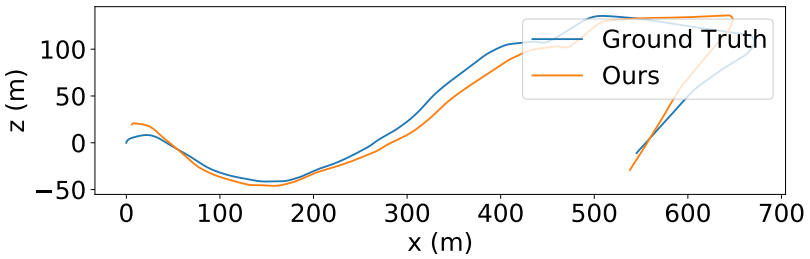

FIGURE 4.16: Trajectory Plot on sequence 10

Structure Pruned SC-Depth

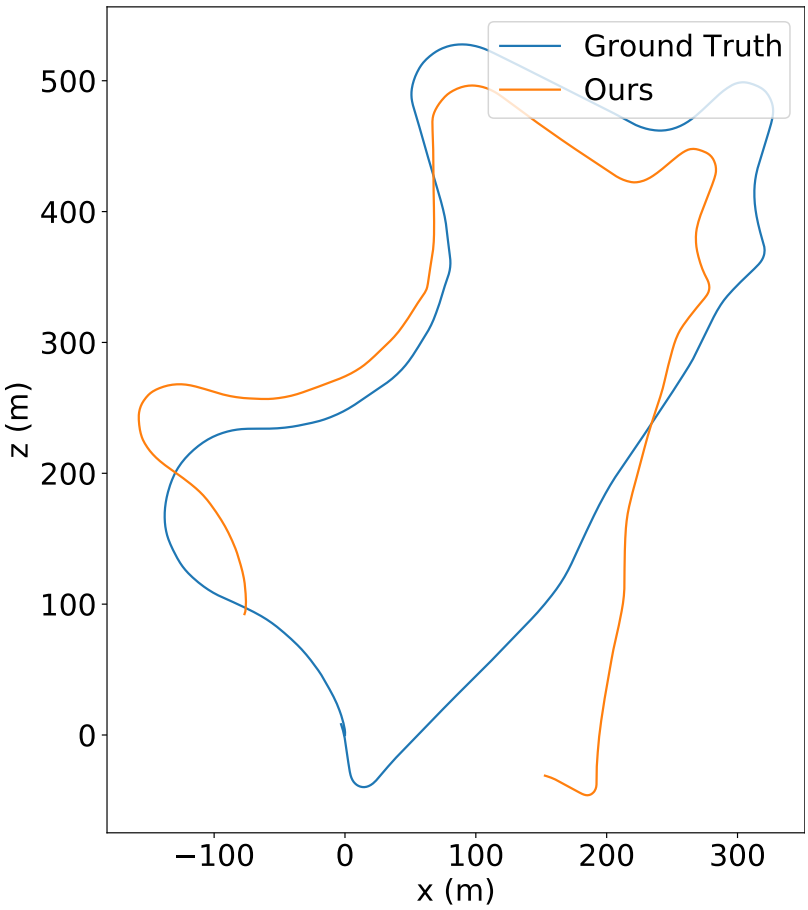

FIGURE 4.17: Trajectory Plot on sequence 09

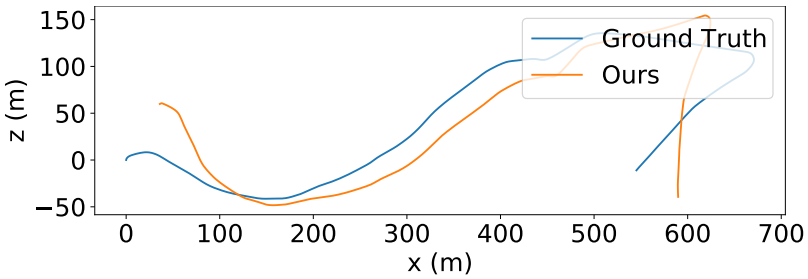

FIGURE 4.18: Trajectory Plot on sequence 10

Pattern Pruned SC-Depth

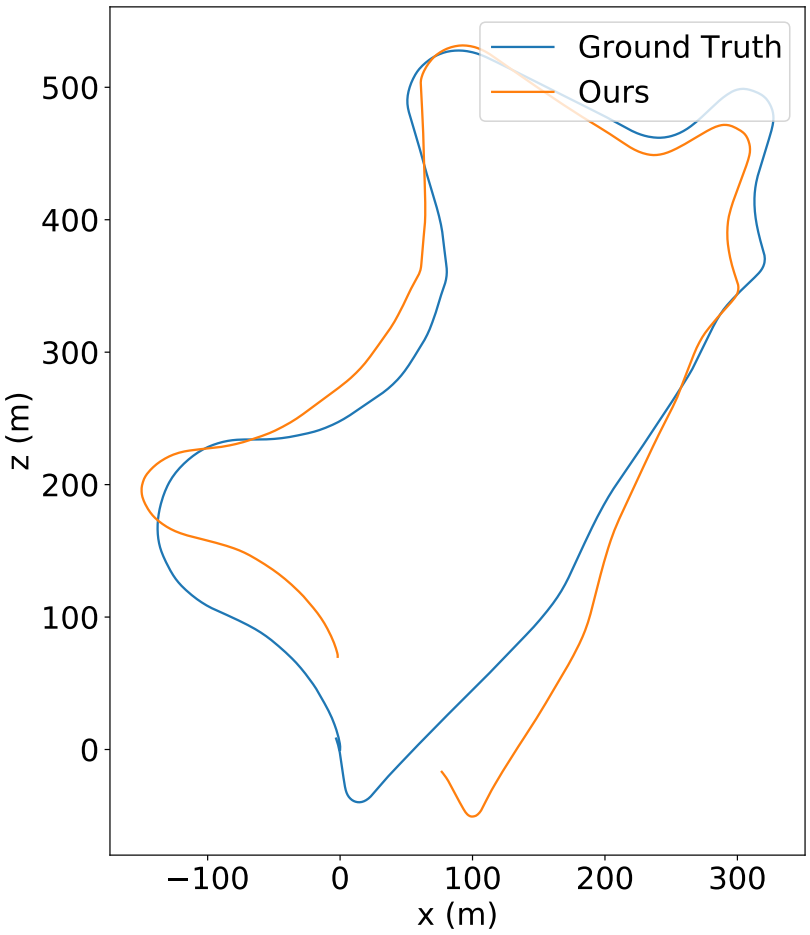

FIGURE 4.19: Trajectory Plot on sequence 09

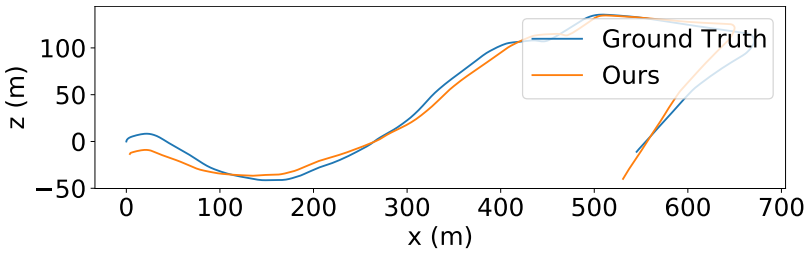

FIGURE 4.20: Trajectory Plot on sequence 10

4.3 Discussion

Theoretically, many datasets have been challenging under the context of computer vision, but each dataset has its focus. As for the visual odometry task, KITTI is the most important dataset to evaluate performance, especially for deep learning frameworks. KITTI provides raw images, and ground truth of camera poses thus is superiority over other datasets of the same category. Therefore, the experiments presented in this section are all based on the dataset KITTI. To make evaluating results more reliable, the translation and rotation errors of normal and pruned SC-Depth presented in Table 4.1 and Table 4.2 are mean values calculated from several experiments.

Camera calibration is a requirement while implementing the SC-Depth, although it can be a drawback compared to DeepVO[8], the much better accuracy and unsupervised learning framework of SC-Depth can easily surpass DeepVO. Additionally, camera calibration is certainly a much simpler task in practical use than collecting the ground truth of camera poses for the training dataset.

As for the pruned SC-Depth, the patterns given in the pattern set are 3×3. Thus only 16 convolution layers can be pruned. According to the patterns given in Fig 3.5, the sparsity of the pruned layers is 55.6%. The patterns are usually given manually so that the shape can influence the performance. Tons of experiments could be done to find an optimal. However, the time cost was unacceptably high at the time, I could only test on a couple of pattern combinations. Implementing reinforcement learning to search for best patterns is also an option, and it could be addressed as future work.

Other pruning strategies like channel pruning and filter pruning can also be added to increase the sparsity further. A combination of different pruning strategies could also be addressed as another future work.

Another benefit of pruning is discovered during the experiments, i.e., the inference process is accelerated. The inference time on testing 2792 images with the original model is 179 seconds, while it only costs 173 seconds for the pruned mode, saving 6 seconds for 2792 images. Although not countable, the computation cost is also saved due to the sparsity in the model. The saved time and computation costs will increase along with sparsity in the model.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

For the specific task related to the soft medical robot given by Inria, if the estimation of the pose of the camera is accurate and applicable, the implementation of the sensor could be safely removed. Thus, both the budget and complexity of implementation could be greatly saved. However, the main challenge is the extremely small size of training dataset. To further promote the project in many other scenarios and make it more general, i.e. when dataset is too small to train a model for visual odometry tasks, how can we improve the accuracy, in this project, an effective pruning strategy is proposed to solve the problem. Moreover, since the model is somehow pruned, the inference process will involve less computation and time costs. Thus, the inference process will be accelerated.

In summary, the primary objective of this study is to improve the accuracy on deep neural network models while training with small datasets on visual odometry tasks and accelerate the inference process.

Visual odometry has been a classic task and has been studied for a long. Deep learning models have been introduced to solve visual odometry tasks until recent years. Thanks to the good accuracy, much more studies have been focusing on improving the deep learning framework these years.

It is not hard to imagine that many datasets are too small to take advantage of the novel algorithms in practical use. However, in this context, the dataset is too small to train a deep neural network and obtain a good prediction. Moreover, it is not the only case for deep neural networks (DNNs) to deal with small datasets. Therefore, this study's main focus is to make such DNNs work for small datasets under the visual odometry context.

One of the most state-of-the-art end-to-end unsupervised learning frameworks is the SC[23]. The advantages of the study state in high accuracy and unsupervised learning framework. Therefore, this graduate project will be extended to the SfMLearner to deal with small datasets.

To operate experiments on small datasets, the training set of KITTI is reduced from 8 sequences to 1 sequence in this project. To make experimental results more general and reliable, experiments are done several times on different training sets of the same size, and the average value is presented in Section *4*.

Pruning is a powerful strategy for compressing a deep model and fitting it with small datasets. However, many different pruning strategies are possible options. As analyzed in Section *3*, pattern pruning is surpassed compared to other pruning strategies. Patterns are usually manually designed and given in a pattern set. In this study, the patterns are in the shape of 3×3. Thus inconsistency with convolution layers and the 16 convolution layers are pruned with a sparsity of 55.6%.

According to the detailed discussion in Section *4*, as a summary, pruning a deep network has benefits as follows:

- Improving accuracy while training on relatively small datasets.
- Accelerating inference process.

5.2 Future Work

As explained in previous section, experiments on INRIA Dataset are temporarily not available since intrinsic parameters of the camera are not known. Fortunately, camera calibration is not a difficult task. The experiments can definitely be done as soon as the camera is reachable.

Moreover, there are also many other pruning strategies that we could use to solve the problem, such as connectivity pruning. Plus, we could also try to combine the different pruning strategies together to achieve an even better result. Therefore, my plan of future work is stated as follows:

- Shooting images of a chessboard with the camera, calibrating the camera and obtain the intrinsic parameters.
- Same as the experiments operated on KITTI dataset, I will operate experiments on INRIA Dataset as well.
- Investigate other pruning strategies and operate experiments.
- Combine different pruning strategies and see if there will be further improvements.

Bibliography

[1] COEVOET E, MORALES-BIEZE T, LARGILLIERE F, **andothers**. Software toolkit for modeling, simulation, and control of soft robots[J/OL]. Advanced Robotics, 2017, 31(22): 1208–1224. eprint: https://doi.org/10.1080/01691864.2017.1395362. https://doi.org/10.1080/01691864.2017.1395362. DOI: 10.1080/01691864.2017.1395362.

[2] HILLER J **and** LIPSON H. Dynamic Simulation of Soft Multimaterial 3D-Printed Objects[J/OL]. Soft Robotics, 2014, 1(1): 88–101. eprint: https://doi.org/10.1089/soro.2013.0010. https://doi.org/10.1089/soro.2013.0010. DOI: 10.1089/soro.2013.0010.

[3] FEI Y **and** XU H. Modeling and Motion Control of a Soft Robot[J]. IEEE Transactions on Industrial Electronics, 2017, 64(2): 1737–1742. DOI: 10.1109/TIE.2016.2572670.

[4] RUNGE G **and** RAATZ A. A framework for the automated design and modelling of soft robotic systems[J/OL]. CIRP Annals, 2017, 66(1): 9–12. https://www.sciencedirect.com/science/article/pii/S000785061730104X. DOI: https://doi.org/10.1016/j.cirp.2017.04.104.

[5] ALLARD J, COTIN S, FAURE F, **andothers**. SOFA - an Open Source Framework for Medical Simulation[C/OL]//Studies in Health Technology and Informatics: MMVR 15 - Medicine Meets Virtual Reality: **volume** 125. Palm Beach, United States: IOP Press, 2007: 13–18. https://hal.inria.fr/inria-00319416.

[6] BIEZE T M, KRUSZEWSKI A, CARREZ B, **andothers**. Design, implementation, and control of a deformable manipulator robot based on a compliant spine[J/OL]. The International Journal of Robotics Research, 2020, 39(14): 1604–1619. eprint: https://doi.org/10.1177/0278364920910487. https://doi.org/10.1177/0278364920910487. DOI: 10.1177/0278364920910487.

[7] BIAN J W, LI Z, WANG N, **andothers**. Unsupervised Scale-consistent Depth and Ego-motion Learning from Monocular Video[J]. ArXiv e-prints, 2019, arXiv:1908.10553: arXiv:1908.10553. arXiv: 1908.10553 [cs.CV].

[8] WANG S, CLARK R, WEN H, **andothers**. DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks[J/OL]. CoRR, 2017, abs/1709.08429. arXiv: 1709.08429. http://arxiv.org/abs/1709.08429.

[9] ZHOU T, BROWN M, SNAVELY N, **andothers**. Unsupervised Learning of Depth and Ego-Motion from Video[J/OL]. CoRR, 2017, abs/1704.07813. arXiv: 1704.07813. http://arxiv.org/abs/1704.07813.

[10] LI R, WANG S, LONG Z, **andothers**. UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning[J/OL]. CoRR, 2017, abs/1709.06841. arXiv: 1709.06841. http://arxiv.org/abs/1709.06841.

[11] ZHAN H, GARG R, WEERASEKERA C S, **andothers**. Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction[J/OL]. CoRR, 2018, abs/1803.03893. arXiv: 1803.03893. http://arxiv.org/abs/1803.03893.

[12] GODARD C, AODHA O M **and** BROSTOW G J. Digging Into Self-Supervised Monocular Depth Estimation[J/OL]. CoRR, 2018, abs/1806.01260. arXiv: 1806.01260. http://arxiv.org/abs/1806.01260.

[13] ALMALIOGLU Y, SAPUTRA M R U, de GUSMAO P P B, **andothers**. GANVO: Unsupervised Deep Monocular Visual Odometry and Depth Estimation with Generative Adversarial Networks[J/OL]. CoRR, 2018, abs/1809.05786. arXiv: 1809.05786. http://arxiv.org/abs/1809.05786.

[14] YANG N, von STUMBERG L, WANG R, **andothers**. D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry[J/OL]. CoRR, 2020, abs/2003.01060. arXiv: 2003.01060. https://arxiv.org/abs/2003.01060.

[15] HAN S, MAO H **and** DALLY W J. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding[J]. ArXiv: Computer Vision and Pattern Recognition, 2016.

[16] ZHANG T, YE S, ZHANG K, **andothers**. A Systematic DNN Weight Pruning Framework using Alternating Direction Method of Multipliers[J/OL]. CoRR, 2018, abs/1804.03294. arXiv: 1804.03294. http://arxiv.org/abs/1804.03294.

[17] REN A, ZHANG T, YE S, **andothers**. ADMM-NN: An Algorithm-Hardware Co-Design Framework of DNNs Using Alternating Direction Methods of Multipliers[C/OL]//ASPLOS '19: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and

Operating Systems. Providence, RI, USA: Association for Computing Machinery, 2019: 925–938. https://doi.org/10.1145/3297858.3304076. DOI: 10.1145/3297858.3304076.

[18] BOYD S, PARIKH N, CHU E, **andothers**. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers[J]. Foundations and Trends in Machine Learning, 2011, 3: 1–122. DOI: 10.1561/2200000016.

[19] WEN W, WU C, WANG Y, **andothers**. Learning Structured Sparsity in Deep Neural Networks[J/OL]. CoRR, 2016, abs/1608.03665. arXiv: 1608.03665. http://arxiv.org/abs/1608.03665.

[20] HE Y, ZHANG X **and** SUN J. Channel Pruning for Accelerating Very Deep Neural Networks[C]//2017 IEEE International Conference on Computer Vision (ICCV). [S.l. : s.n.], 2017: 1398–1406. DOI: 10.1109/ICCV.2017.155.

[21] MAIRAL J, KONIUSZ P, HARCHAOUI Z, **andothers**. Convolutional Kernel Networks[C/OL]//GHAHRAMANI Z, WELLING M, CORTES C, **andothers**. Advances in Neural Information Processing Systems: **volume** 27. [S.l.]: Curran Associates, Inc., 2014. https://proceedings.neurips.cc/paper/2014/file/81ca0262c82e712e50c580c032d99b60-Paper.pdf.

[22] ZHANG R. Making Convolutional Networks Shift-Invariant Again[J/OL]. CoRR, 2019, abs/1904.11486. arXiv: 1904.11486. http://arxiv.org/abs/1904.11486.

[23] BIAN J, LI Z, WANG N, **andothers**. Unsupervised Scale-consistent Depth and Ego-motion Learning from Monocular Video[J/OL]. CoRR, 2019, abs/1908.10553. arXiv: 1908.10553. http://arxiv.org/abs/1908.10553.

[24] ZHAO W, LIU S, SHU Y, **andothers**. Towards Better Generalization: Joint Depth-Pose Learning without PoseNet[J/OL]. CoRR, 2020, abs/2004.01314. arXiv: 2004.01314. https://arxiv.org/abs/2004.01314.

[25] JADERBERG M, SIMONYAN K, ZISSERMAN A, **andothers**. Spatial Transformer Networks[C/OL]//CORTES C, LAWRENCE N, LEE D, **andothers**. Advances in Neural Information Processing Systems: **volume** 28. [S.l.]: Curran Associates, Inc., 2015. https://proceedings.neurips.cc/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf.

[26] WANG Z, BOVIK A, SHEIKH H, **andothers**. Image quality assessment: from error visibility to structural similarity[J]. IEEE Transactions on Image Processing, 2004, 13(4): 600–612. DOI: 10.1109/TIP.2003.819861.

[27] RANJAN A, JAMPANI V, KIM K, **andothers**. Adversarial Collaboration: Joint Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation[J/OL]. CoRR, 2018, abs/1805.09806. arXiv: 1805.09806. http://arxiv.org/abs/1805.09806.

[28] YIN Z **and** SHI J. GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose[Z]. 2018. arXiv: 1803.02276 [cs.CV].

[29] WANG Y, WANG P, YANG Z, **andothers**. UnOS: Unified Unsupervised Optical-Flow and Stereo-Depth Estimation by Watching Videos[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). [S.l. : s.n.], 2019.

[30] LI Z, YUAN G, NIU W, **andothers**. 6.7ms on Mobile with over 78% ImageNet Accuracy: Unified Network Pruning and Architecture Search for Beyond Real-Time Mobile Acceleration[J/OL]. CoRR, 2020, abs/2012.00596. arXiv: 2012.00596. https://arxiv.org/abs/2012.00596.

[31] MA X, GUO F M, NIU W, **andothers**. PCONV: The Missing but Desirable Sparsity in DNN Weight Pruning for Real-time Execution on Mobile Devices[Z]. 2020. arXiv: 1909.05073 [cs.LG].

[32] YAMINS D **and** DICARLO J J. Using goal-driven deep learning models to understand sensory cortex[J]. Nature Neuroscience, 2016, 19: 356–365.

[33] YUAN G, BEHNAM P, LI Z, **andothers**. FORMS: Fine-grained Polarized ReRAM-based In-situ Computation for Mixed-signal DNN Accelerator[J/OL]. CoRR, 2021, abs/2106.09144. arXiv: 2106.09144. https://arxiv.org/abs/2106.09144.

[34] REN A, ZHANG T, YE S, **andothers**. ADMM-NN: An Algorithm-Hardware Co-Design Framework of DNNs Using Alternating Direction Method of Multipliers[J/OL]. CoRR, 2018, abs/1812.11677. arXiv: 1812.11677. http://arxiv.org/abs/1812.11677.

[35] KINGMA D P **and** BA J. Adam: A Method for Stochastic Optimization[Z]. 2017. arXiv: 1412.6980 [cs.LG].

[36] BOJNORDI M N **and** IPEK E. Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning[C]//2016 IEEE International Symposium on High Performance Computer Architecture (HPCA). [S.l. : s.n.], 2016: 1–13. DOI: 10.1109/HPCA.2016.7446049.

[37] CORDTS M, OMRAN M, RAMOS S, **andothers**. The Cityscapes Dataset for Semantic Urban Scene Understanding[J/OL]. CoRR, 2016, abs/1604.01685. arXiv: 1604.01685. http://arxiv.org/abs/1604.01685.

[38] GEIGER A, LENZ P, STILLER C, **andothers**. Vision meets robotics: The KITTI dataset[J/OL]. The International Journal of Robotics Research, 2013, 32(11): 1231–1237. eprint: https://doi.org/10.1177/0278364913491297. https://doi.org/10.1177/0278364913491297. DOI: 10.1177/0278364913491297.

[39] ZHOU T, BROWN M, SNAVELY N, **andothers**. Unsupervised Learning of Depth and Ego-Motion from Video[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l. : s.n.], 2017: 6612–6619. DOI: 10.1109/CVPR.2017.700.