ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

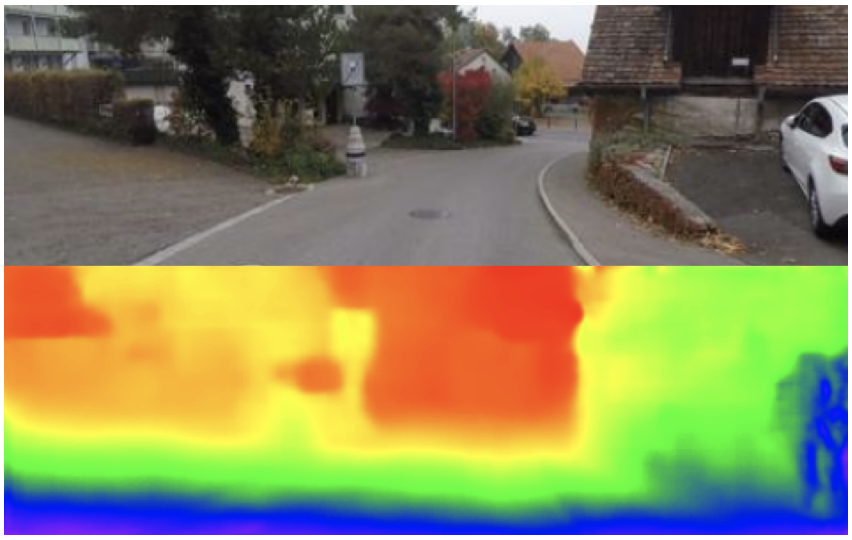
Institut für Geodäsie und Photogrammetrie
Institute of Geodesy and Photogrammetry

Institute of Geodesy and Photogrammetry
& Computer Vision Laboratory

Autumn Semester 2018

Semi-supervised Learning of Depth from Video

Interdisciplinary Project



Qi Dai
daiq@student.ethz.ch

January 2019

Supervisors: Simon Hecker, heckers@vision.ee.ethz.ch
Dengxin Dai, dai@vision.ee.ethz.ch

Professors: Prof. Dr. Konrad Schindler
Prof. Dr. Luc Van Gool

Acknowledgements

I really appreciate **Simon Hecker** a lot for the instruction and help throughout the whole project. I would like to thank **Prof. Dr. Konrad Schindler**, **Prof. Dr. Luc van Gool** and **Dr. Dai Dengxin** a lot for offering me the opportunity of working on this project. I also want to express my gratitude for the support from the Computer Vision Laboratory, ETH Zurich.

Abstract

Learning-based monocular depth estimation has shown promising results. In particular, the unsupervised approach based on view synthesis has achieved comparable or even better performance than the supervised method. Two networks are jointly trained to predict the depth map and ego-motion. However, there is a scale ambiguity of the depth prediction. The output depth value are just up-to-scale.

In this project, we present a semi-supervised learning framework which can directly give the absolute depth map. The scale ambiguity is removed through introducing a new loss term. We evaluate our framework on both KITTI and our own dataset, and the result shows that the absolute loss term is able to upgrade the model from relative to absolute version.

Contents

1. Introduction

Depth estimation from image has been researched for a long time. Depth map can also be used in many scenarios. For example, it can provide useful information for driving an autonomous vehicles. The distance information may reduce the risk of collision and help build a safer autonomous driving system. Depth map is also a good alternative to the Lidar, which is more expensive and cumbersome to carry.

Mathematically, the depth can be computed given an stereo image pair (at least two images). It seems impossible to infer the depth when only a single image is given, since the depth information is lost during the 3D (object in world) to 2D (object on image) projection.

However, it's obvious that we can perceive the image depth when looking at a single image. Humans can exploit their previous knowledge to infer the image depth, like the scales relative to the known size of familiar objects. In short, monocular depth estimation is possible in intuition.

Due to the importance of monocular depth estimation, this task has been extensively researched in recent years. Like other computer vision tasks, approaches based on deep learning achieve remarkable success. Earlier works cast this task as a supervised learning problems[1][2][3]. In these methods the models have been trained with a large dataset, where the depth ground truth is available and used for supervising the prediction. However, high quality depth ground truths are hard to obtain. These annotations are often collected by the expensive laser or depth camera.

Besides supervised method, recent works[4][5][6] shows that *view synthesis* can be an effective supervisory signal for depth prediction. The idea is to apply the photometric consistency on the synthesized view, while the photometric difference provides a supervisory signal for training the model. In this method no ground truth is needed, while its prediction performance is still comparable or even superior to its supervised counterpart.

In this project, we will exploit the monocular depth estimation based on *view synthesis*. Our framework is adapted from the architecture proposed by Zhou *et al.*[4], where a relative depth map can be predicted. We try to upgrade the framework to output absolute depth value, then train and evaluate on different datasets.

1.1. Related work

Monocular depth estimation can be divided into supervised or unsupervised approach, depending on whether the depth ground truth is needed during training process.

Supervised monocular depth estimation Most supervised approaches are formulated as a regression problem, where the difference between the depth prediction and ground truth is minimized. In early work the feature is manually defined. Saxena *et al.*[7] propose to estimate the single-view depth by training Markov random field(MRF) with hand-crafted features. In [8] Liu *et al.* integrate semantic labels with MRF learning. Ladicky *et al.* improve the depth estimation performance by combining the semantic labeling with the depth estimation.

The success of deep learning inspires many other methods. Eigen *et al.*[1] propose a deep convolutional neural network(CNN) architecture to produce dense pixel depth. Based on this architecture, many variant structures have been proposed to improve the prediction performance. Li *et al.*[9] improve the estimation accuracy by combining the CNNs with the conditional random filed(CRF), while Laina *et al.*[10] use the more robust *Huber loss* as the loss function. Cao *et al.*[11] formulate the depth prediction problem as a pixel-wise classification task. Besides CNNs, Kumar *et al.*[12] propose a model with recurrent neural network(RNN) to provide spatio-temporally accurate monocular depth estimation.

However, supervised approach requires large amounts of of accurate and pixel aligned depth ground truth. And ideally, various scenes should be covered in the training dataset. This type of data is difficult to obtain in real world. The lack of ground truth depth map may hinder the performance of supervised approach. One possible solution is to use syntheic data with perfect depth ground truth. In recent work [13], synthetic images with highly accurate depth ground truth are use to train a depth estimation network, and an image style transfer network is trained to convert a real image into the synthetic domain, then the depth map can be estimated from real image.

Unsupervised monocular depth estimation The photometric consistency between nearby frames makes it possible to predict the depth without the ground truth. One category is to learn depth from stereo image pairs, where the pose between the stereo cameras is already known. Grag *et al.*[14] trains a network to predict the depth that minimize the photometric difference between the true right view and the synthesized right view. They use Taylor expansion to approximate the cost function and derive the gradient. As a result this approximated objective is only sub-optimal. Godard *et al.*[5] choose the spatial transformer network, yielding a cost function which is differentiable without any approximation. They reconstruct the image using the predicted disparity, where the left-right consistency consistency is enforced to encourage a more accurate prediction.

Another category is to infer from single image sequence. Vijayanarasimhan [15] and Zhou *et al.* [4]show that the learning of depth prediction and ego-motion at the same time is possible. The depth network and pose network are jointly trained, and the supervision signal is introduced by minimizing the photometric difference between the synthesized view and true view. The ego-motion estimation from pose network makes it possible to train without the stereo image pairs. In recent works, several other constraints are introduced during the training. Yang *et al.* [16] utilize the consistency between normal and depth. Mahjourian *et al.* [17] propose a 3D point cloud alignment loss.

1.2. Main tasks

In my project I mainly focus on the following two aspects:

Absolute upgrade The framework proposed by Zhou *et al.*[4] can only provide relative depth. This is because the whole framework is trained in an unsupervised way, no absolute information is used in this process. The predicted depth can be scaled arbitrarily without increasing the predefined loss.

Although there is only a scaling difference between the absolute depth map and the relative prediction, in reality this scaling factor is hard to find. In our project, we encourage the network to output absolute depth by introducing an absolute loss term L_{abs}. The framework structure is modified to integrate the L_{abs} with previously defined loss.

In short, we formulate a new loss function, then train and evaluate on this modified framework. More details regarding the absolute loss are in section 2.3.

Evaluate on our own dataset Besides the widely used KITTI dataset, we also format our own dataset to do some training and evaluation. The image data is collected by Gopro Hero5, in different areas of Switzerland.

Some prepossessing steps are needed to use these images. We conduct single camera calibration to compute camera intrinsic, stereo calibration to determine the relationship between different cameras. We also generate some ground truth of depth map for evaluation.

We train and evaluate our framework on both KITTI and Gopro dataset, and compare the results with each other. More details about the Gopro dataset can be found in section 3.2.

1.3. Report structure

This report is organized as following:

Section 2 illustrates the background theory of *view synthesis*, which is the main supervision signal of the framework. The formulations of different loss term are presented and explained, including the appearance loss L_{ap}, the smooth loss L_{smooth} and the absolute loss L_{abs}. This section ends by detailing the architecture of two networks used in our framework: Depth-net and Pose-net.

In Section 3 the description of datasets and some prepossessing steps are given. We first present how KITTI dataset is used in our project, including the images we used and how we split the train and test part. Then we give the details of Gopro dataset, like the platform setup, the conversion from raw data, etc. The generation of depth ground truth is illustrated in the end of this section.

The results are given and analyzed in Section 4. We present the qualitative results on KITTI and Gopro dataset respectively, following some numerical evaluations for the comparison of the prediction accuracy. Some analysis regarding the results are also given.

This report is consummated with a summary of the project in Section 5, where some conclusions and possible future work are also presented.

2. Method

In the project we focus on the results of depth prediction, while the adopted framework can provide both depth and ego-motion estimation. Figure 2.1 illustrates the general structure.

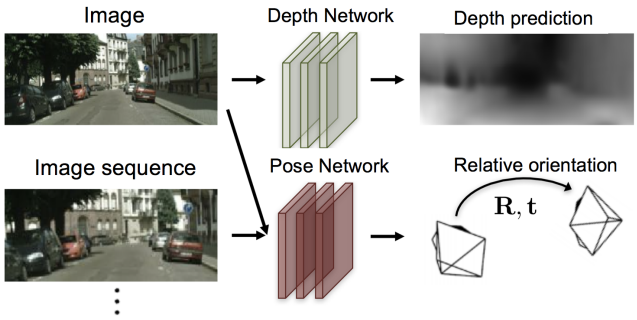


Figure 2.1.: Framework Overview, adapted from Zhou et al. (2017)

There are two CNNs in this framework. One is the *Depth network* (Depth-net) which takes a single image as input, then output a pixel-wise depth map($\hat{D}$). The other is the *Pose network* (Pose-net), feeding with image sequence, and output the relative camera pose($\hat{T}$) between different scenes. Here the image sequence consists of scenes captured by a moving camera.

These two networks must be jointly trained, but can be tested independently. This is because to synthesize the image (details in section 2.1), the results from both networks are needed.

2.1. View synthesis as supervision

As mentioned before, the Depth-net is trained in an unsupervised way, no depth ground truth is used. The main supervision signal comes from the appearance difference between the synthesized view and the true target view.

View synthesis is to synthesize an view (denoted as target view: I_t) from a given image (denoted as source view: I_s). I_t and I_s should be the same scene but captured at different perspective. To synthesize the view, it is necessary to compute the relative camera pose

$\hat{T}_{t\to s}$ between the moving camera captured I_t and I_s, as well as the depth map $\hat{D}_t$ for the target view I_t.

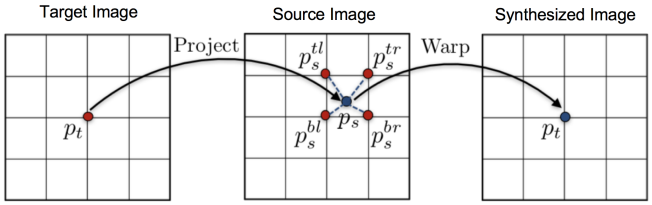


Figure 2.2.: View synthesis, adapted from Zhou et al. (2017)

As illustrated in Figure 2.2, the synthesized image is reconstructed pixel by pixel. In other word, for each pixel p_t on the target image I_t, we need to find its corresponding pixel p_s on source image I_s. This correspondence relationship satisfies the following formula:

$$p_s \sim K\hat{T}_{t\to s}\hat{D}_t(p_t)K^{-1}p_t \tag{2.1}$$

p_t and p_s are homogeneous coordinates of a specific pixel on I_t and I_s respectively. K denotes the camera intrinsic matrix. $\hat{T}_{t\to s}$ is a 3×3 matrix, describing the relative camera pose. $\hat{D}_t(p_t)$ is a scalar value denoting the depth for p_t.

However, the computed p_s is a continuous value, lying between different pixels of the source image I_t. To sample the pixel intensity of p_t from I_t, the bi-linear sampling method is adopted to interpolate the value. The intensity of four neighboring pixels around p_s are used (top-left, top-right, bottom-left and bottom-right). The final pixel intensity is interpolated by:

$$\hat{I}_s(p_t) = I_s(p_s) = \sum_{i\in\{t,b\},j\in\{l,r\}} w^{ij} I_s(p_s^{ij}) \tag{2.2}$$

where w_{ij} is linearly proportional to the spatial proximity between p_s and p_s^{ij}.

The use of bi-linear interpolation is also locally fully differentiable, considering the back-propagation of gradient during the training process. This means neither simplification nor approximation is needed for the cost function.

Here we introduce our first loss term: *appearance loss* L_{ap}, which can be formulated as:

$$L_{ap} = \frac{1}{N}\sum_{i,j} |(I_t - \hat{I}_s)^{ij}| \tag{2.3}$$

where I_t is the true target view, and $\hat{I}_t$ is the synthesized view from source image I_s. The final L_{ap} equals to the average of pixel intensity difference over all image pixels.

2.2. Multi-scale depth prediction & smooth regularization

The gradient derived from L_{ap} is proportional to the pixel intensity difference between $I_t(p_t)$ and four neighbors of $I_s(p_s)$. If the correct p_s is located in texture-less area or far from the correct estimation, it will be difficult to train, since the gradient is either close to zero or slow to converge.

To overcome this problem, the multi-scale appearance loss is applied. The target image will be synthesized at four different scales, which double in image size at each of the subsequent scales. The final appearance loss equals to the sum of loss over all scales:

$$L_{ap} = \sum_{scale} L_{ap,scale} \tag{2.4}$$

The depth map at different scales is also needed. With the introduction of multi-scale depth estimation, the gradient can be derived from a larger spatial context, which facilitates the back-propagation of gradient even in texture-less area and accelerate the training speed.

We also expect the depth map prediction to be smooth. An real depth map of a scene should be smooth in most parts, while only have a sudden depth change at area where two objects are spatially separated.

We apply the $L1$ regularization on the predicted depth map. This is because the L1 penalization tends to result in a sparse distribution of smoothness: where only a small proportion of area is allowed to be fluctuated, while most part are smooth(close to zero).

In our implementation, the $L1$ norm of the second-order gradients for the disparity map should be minimized. Here we apply smoothness regularization on the predicted disparity, instead of the depth map, because the direct output of the Depth-net is a disparity. More details regarding the Depth-net is given in 2.4.

This smoothness regularization is imposed over all involved scales. We apply a correction factor for disparity smoothness at different scales as shown in the following equation:

$$L_{smooth} = \sum_{scale} \frac{1}{2^{scale}} \frac{1}{N_{scale}} \sum_{i,j} |\partial_{xx}\hat{D}^{ij}_{scale}| + |\partial_{xy}\hat{D}^{ij}_{scale}| + |\partial_{yx}\hat{D}^{ij}_{scale}| + |\partial_{yy}\hat{D}^{ij}_{scale}| \tag{2.5}$$

Here the *scale* ranges from 0 to 4, corresponding to the 4 disparity maps the Depth-net predicts. $\frac{1}{2^{scale}}$ is the correction factor, while $\frac{1}{N_{scale}}$ is to average the smoothness over all image pixels.

2.3. Absolute-loss term

Until now no absolute information is introduced in our framework. Thus the network output, $\hat{D}$ and $\hat{T}$ from Depth-net and Pose-net are just relative depth and translation. This is because the synthesized view are computed according to equation 2.1. As no constraint (absolute information) is imposed, the predicted depth or translation elements in $\hat{T}$ can be scaled arbitrarily, as long as these two terms can jointly conform the relationship indicated in equation 2.1.

To upgrade the framework to output true absolute value, we introduce an *absolute loss* term L_{abs} on the translation output from Pose-net:

$$L_{abs} = |t_{true} - t_{pred}| \tag{2.6}$$

where t_{true} and t_{pred} are the true and the prediction of the camera translations respectively. $t = (t_x, t_y, t_z)$, while the movement is mainly along the z-axis, as the vehicle often drive forwardly.

The true translation is inferred from the vehicle speed. As the time interval between the capture of two frames is rather small(normally only 0.1s), we assume the vehicle experience uniform acceleration linear motion during this period. The speed is recorded by the *inertial navigation system*(INS) in KITTI dataset, or CAN-Bus in Gopro dataset.

The L_{abs} encourages the Pose-net to predict the true absolute translations, and thus forces the Depth-net to give absolute depth accordingly. Notice that the L_{abs} is irrelevant with different scales. This is because L_{abs} is imposed on the predicted camera pose, which remains constant even we re-scale the captured image.

The final loss is a weighted sum of the three loss mentioned above, as illustrated in Figure 2.3. The objective function can be formulated as equation 2.7.

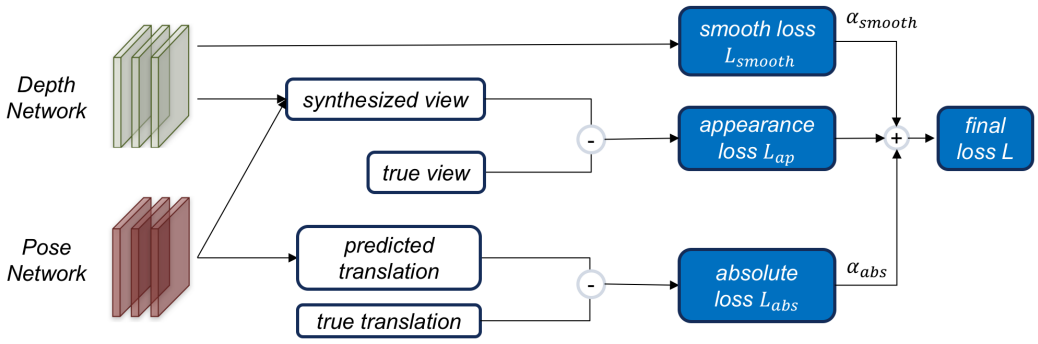


Figure 2.3.: Illustration of final loss, adapted from Zhou *et al.* (2017)

$$L = L_{ap} + \alpha_{smooth} L_{smooth} + \alpha_{abs} L_{abs} \tag{2.7}$$

Here two weight coefficients α_{smooth} or α_{abs} should be determined before training.

2.4. Network architecture

Depth-net The architecture of Depth-net is illustrated in Figure 2.4. This network is first proposed by *Mayer et al.*[18], where an Encoder-Decoder structure is adopted. The size of the predicted depth map is guaranteed to be the same with the one of input image.

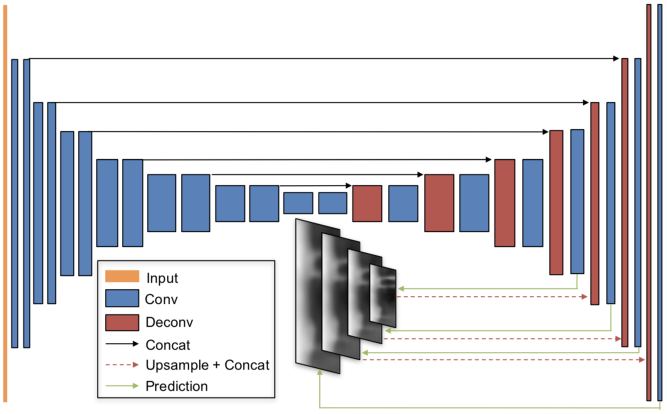


Figure 2.4.: Depth-net, from Zhou *et al.* (2017)

The input of Depth-net is a single RGB image, and output depth map at four different scales. As shown in Figure 2.4, the decoder adopts skip connection by concatenating feature maps from encoder, and output multi-scale side predictions. Each convolutional layer is followed by ReLU activation, except for those prediction layers. The activation for prediction layer is *sigmoid* function whose output ranges from 0 to 1. We re-scale this output and add a bias term to get the predicted disparity map:

$$disp = \alpha \times sigmoid(x) + \beta \tag{2.8}$$

So the direct output of Depth-net is actually the disparity. To compute the depth from the disparity prediction, a transformation is needed, as shown in the following formula:

$$depth = \frac{1}{disp} \tag{2.9}$$

For a rectified stereo image pair, the transformation between disparity and depth is: $depth = bf/disp$, where b and f are the baseline length (in unit of meter) and focal length (in unit of pixel). Compared with equation 2.9, it can be inferred that we assume

there is a virtual cameras besides the real camera, whose product of the baseline and focal length is 1. While the predicted disparity is with respect to the frame captured by this virtual camera.

Meanwhile the α and β in equation 2.8 reflects our belief for the depth range. We choose $\alpha = 10$ and $\beta = 0.007$, meaning that the depth ranges from 0.01 to around 143 meters.

Table 2.1.: Details of Depth-net

Encoder Part				
layer	**k**	**s**	**channels**	**input**
conv1	7	2	3/32	RGB
conv1b	7	1	32/32	conv1
conv2	5	2	32/64	conv1b
conv2b	5	1	64/64	conv2
conv3	3	2	64/128	conv2b
conv3b	3	1	128/128	conv3
conv4	3	2	128/256	conv3b
conv4b	3	1	256/256	conv4
conv5	3	2	256/512	conv4b
conv5b	3	1	512/512	conv5
conv6	3	2	512/512	conv5b
conv6b	3	1	512/512	conv6
conv7	3	2	512/512	conv6b
conv7b	3	1	512/512	conv7

Decoder Part				
layer	**k**	**s**	**channels**	**input**
upconv7	3	2	512/512	conv7b
iconv7	3	1	1024/512	upconv7+conv6b
upconv6	3	2	512/512	iconv7
iconv6	3	1	1024/512	upconv6+conv5b
upconv5	3	2	512/256	iconv6
iconv5	3	1	512/256	upconv5+conv4b
upconv4	3	2	256/128	iconv5
iconv4	3	1	128/128	upconv4+conv3b
disp4	3	1	128/1	iconv4
upconv3	3	2	128/64	iconv4
iconv3	3	1	129/64	upconv3+conv2b+*disp4*
disp3	3	1	64/1	icon3
upconv2	3	2	64/32	iconv3
iconv2	3	1	65/32	upconv2+conv1b+*disp*
disp2	3	1	32/1	iconv4
upconv4	3	2	256/128	iconv5
iconv4	3	1	128/128	upconv4+conv3b
disp4	3	1	128/1	iconv4

More details regarding the Depth-net can be found in the above Table 2.1. Here the **k** and **s** are short for kernel and stride respectively. In the **channel** column the number of input and output channel is given.

Pose-net The input of Pose-net is an image sequence consists of source view I_s and target view I_t. They are concatenated along the color channel. The network output is the relative camera pose, parameterized as a 6-dimensional vector (corresponding to 3-D translation and 3 Euler angles). Suppose the image sequence contains N images, and only one of them is the target view we want to synthesize. Then the network will output $6 \times (N-1)$ values for each source view in this sequence.

The Pose-net has 7 stride-2 convolutional layers, followed by one 1×1 convolutional layer which gives the pose prediction. All convolutional layers in the network is followed by ReLU activation except for the last prediction layer, where no non-linear activation is

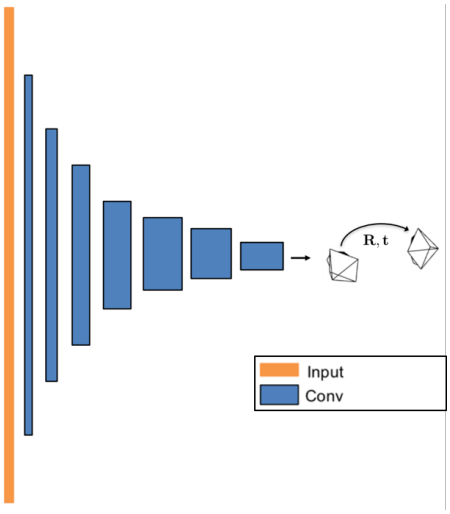


Figure 2.5.: Pose-net, adopted from Zhou *et al.* (2017)

applied. More details of the Pose-net can be found in Table 2.2. The N in this table is the number of images in the image sequence.

Table 2.2.: Details of Pose-net

Encoder Part				
layer	**k**	**s**	**channels**	**input**
conv1	7	2	$[3 \times N]/16$	RGB
conv2	5	2	16/32	conv1
conv3	3	2	32/64	conv2
conv4	3	2	64/128	conv3
conv5	3	2	128/256	conv4
conv6	3	2	256/256	conv5
conv7	3	2	256/256	conv6
pred	1	1	$256/[6 \times (N-1)]$	conv7

3. Data

We train and evaluate the above framework on two datasets. One is the popular KITTI dataset[19], the other is the Gopro dataset which is collected by ourselves. In this chapter the general description of the datasets is given, and some prepossessing steps are explained.

3.1. KITTI dataset

KITTI dataset is widley used in the field of autonomous driving. Its platform consists of multiple sensors, including one inertial navigation system (GPS/IMU), one laser scanner, two grayscale and two color cameras, etc. All sensors have been calibrated with each other, and the captured images have also been synchronized with other data.

In our implementation only the color images are used. The camera intrinsic matrix is provided. The true translation value between I_t and I_s, which is corresponding to the output of Pose-net, is computed from the records of inertial navigation system.

The static frames in dataset should be excluded. This is because the *view synthesis* is used as the main supervision signal, while it is meaningless to synthesize a target view which is exactly the same with the input source view. After excluding static frames, finally 40109 frame pairs are used for training.

For the evaluation of depth prediction, we use the 697 images from Eigen test split[1]. The depth ground truth are generated from the output of laser scanner. As the scanner only provides the coordinates of laser point, the ground truth depth map will be relatively sparse. In testing step we only compare pixels with depth value, no interpolation is applied to fill the missing depth.

3.2. Gopro dataset

We also collect some data by ourselves. The recording platform is equipped with various sensors, including multiple Gopro cameras which provide a panoramic view of the surrounding, the Bumblebee stereo cameras for generating depth ground truth, a CAN bus for recording speed, etc.

In our project mainly the following data are used for training:

1. Front-view RGB images collected by Gopro Hero5;
2. Vehicle speed from CAN bus (for computing the translation);

Originally the images are captured in 60Hz as a video. We re-sample this video in 10Hz and synchronize them with other data. The collected frames cover various scenes, like city area, countryside, residential part and expressway, etc. We only use frames in city and residential area for training and testing, since frames in these areas tend to have more texture, which may benefit the network performance a lot.

The true translation value between frames is computed from the product of vehicle speed and time interval between the capture of images. We regards the time interval as 0.1s, since our sample rate is 10 Hz.

The camera intrinsic matrix is needed for both view synthesis (see equation 2.1) and stereo camera calibration (in section 3.2.1). So both the Gopro Hero5 and Bumblebee camera is calibrated in advance.

3.2.1. Depth ground truth generation

The ground truth depth maps are required for the evaluation of the Depth-net performance. Our platform is equipped with Bumblebee XB3, a stereo vision camera system which can be used to generate depth ground truth. 3 calibrated cameras are rigidly mounted on the Bumblebee with known baseline. We use two of them to capture the RGB images of the scene, and compute the depth map through pipeline of 3D reconstruction.

Figure 3.1.: RGB image (captured by the left Bumblebee camera)

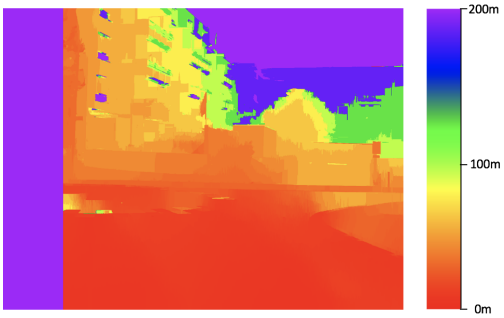


Figure 3.2.: Depth map

Figure 3.1 and 3.2 shows the RGB image captured by the (left) Bumblebee camera and its corresponding depth map. For the sake of visualization, we assign all pixels whose depth are larger than 200 as 200 meters.

It can be seen that the generated depth map is not perfect, with some inaccurate depth in area like the windows or the circular crossroad on the ground. It is true that the ground

truth generated from Bumblebee still contains some errors, although the proportion of inaccuracy is rather small.

Note that the depth values at the left border are missing. This is because pixels at the left border have no correspondence on the image captured by the right camera.

The generation of the depth ground truth consists of the following steps:

1. Generation of the depth map in the Bumblebee camera coordinate system;
2. Stereo calibration to get the relative pose between the Gopro camera and the Bumblebee camera;
3. Transformation of the depth into the Gopro camera coordinate system.

The transformation of the depth from the Bumblebee camera coordinate system to Gopro coordinate system is needed, since the Depth-net only provides the depth map in the latter system.

We use D_b and D_g to denote the depth map in Bumblebee and Gopro coordinate system respectively. The rotation matrix $R_{b\to g}$ and translation vector $T_{b\to g}$ are inferred through stereo camera calibration. Then we back-project each pixel on D_b into a 3D point (say X_b) inside the Bumblebee system, and apply the transformation according to formula 3.2.1 to get the 3D point X_g inside the Gopro system.

$$X_g = R_{b\to g}X_b + T_{b\to g}$$

Finally the desired depth map is computed through projecting these 3D points onto the image plane given the Gopro camera intrinsic matrix. The result is shown in Figure. 3.4.

Figure 3.3.: Original RGB image

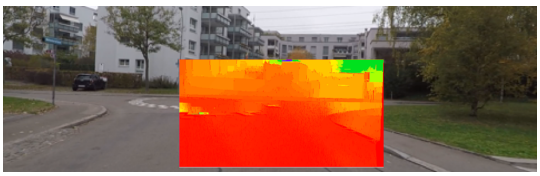


Figure 3.4.: Ground truth depth map on Gopro image plane

Note that we ignore the sky area in the ground truth generation, the same as what Eigen *et al.*[1] does in .

It can be seen that the depth values of a large proportion of pixels are missing, due to the difference of field of view between the Bumblebee camera and the Gopro camera. In evaluation we only consider pixels with ground truth.

4. Experiments & Results

We evaluate our framework on KITTI and Gopro datasets. The network is implemented in Tensorflow[20] with more than 33 million trainable parameters. The implementation details and evaluation results are given in the following part.

4.1. KITTI dataset

The input of Pose-net should be an image sequence. The number of images in the sequence is set as 3 for both experiments. For a sequence, we treat the central image as the target view, which we try to synthesize from the other two images(source view). We resize the image to 128×416 during training. While the original image size is 1392×512

The training takes about 13 hours on a single Titan X GPU for 20 epochs, with the mini-batch size as 4. We set $\alpha_{smooth} = 0.2/s$(s is the scaling factor for the multi-scale disparity output), $\alpha_{abs} = 1$. The Adam optimizer is used with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. And the learning rate is fixed to be 0.0002 throughout the training process.

Figure 4.1 shows some examples of depth prediction. It can be seen that the general structure of the image has been captured correctly.

Table 4.1 gives the numerical evaluation of the depth prediction. We compare the performance between the result from Zhou *et al.* and us. For the first three indicators,a smaller value indicates a better performance. For the last three columns a higher value is good. The details of these numerical indicators are given in the appendix A.

Table 4.1.: Numerical result on KITTI

	Abs Rel	Sq Rel	RMSE	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original (relative)	0.1882	1.6386	6.7377	0.7259	0.8981	0.9574
Ours (absolute)	0.3841	8.4528	9.7858	0.5645	0.8021	0.9059

Since the original results are just up-to-scale, for evaluation these prediction values are multiplied with a scaling factor $\hat{s} = median(D_{gt})/median(D_{pred})$, which matches with the median of the ground truth. While for our result no scaling is needed.

It can be seen that our results doesn't perform as well as the original one. By checking the result, we found the absolute loss L_{abs} is a little bit high (shown in Figure 4.2).

Figure 4.1.: Depth prediction(KITTI dataset)

The absolute loss drops fast at beginning while keeps constant (around 0.4) in the following training steps. As presented in section 2.3, the L_{abs} equals to the difference between the true translation and its prediction. On average the translation between each frame is around 1 meter. So this means the predicted translation is around 0.6 meter, which is always 0.4 less than the true value.

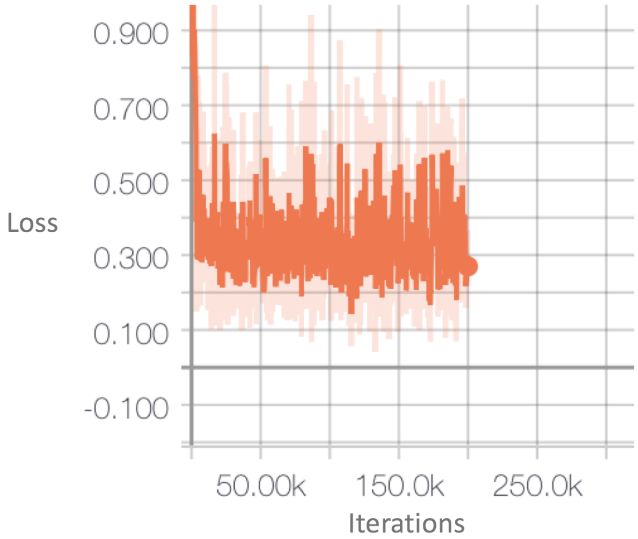


Figure 4.2.: Absolute loss L_{abs} on KITTI dataset

Thus we think the training result only achieves a sub-optimum. Currently we doesn't manage to further improve the performance. This will be done in the future work, through tuning some hyper-parameters or introducing more constraints.

4.2. Gopro datasets

In total 62597 image sequences are formatted for training. They are mainly captured in the city or the residential area, where more texture can be found. To facilitate training we crop the sky part and the bottom part(the front end of vehicle) of the image, and the final image size is 149×480.

We train the network from scratch for 20 epochs, taking about 18 hours on a single Titan X GPU. The mini-batch size is 15. The smooth weight $\alpha_{smooth} = 0.2/s$, absolute weight $\alpha_{abs} = 0.07$. All other parameter settings are same with the one of KITTI experiment.

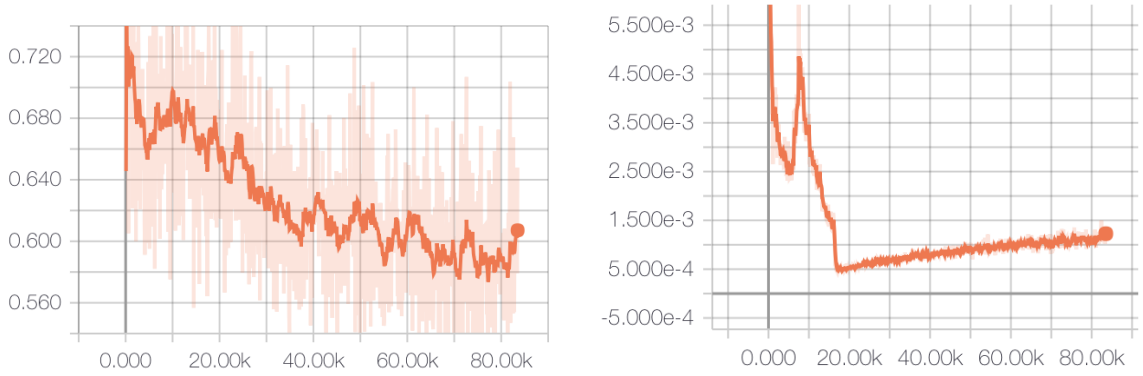
Our depth prediction has a good performance on textured area, but tends to fails for homogeneous areas. As shown in the example of the third row of Figure 4.3, the network inaccurately predicts a large depth value for the nearby ground.

Figure 4.3.: Depth prediction(Gopro dataset)

The inaccurate depth prediction in texture-less area results from the fact that, when synthesizing the view, an inaccurate depth in homogeneous part won't result in a big pixel intensity difference, since all nearby pixels have similar intensity.

Intuitively, the introduce of smoothness constraint on disparity map may mitigate this issue, since the it penalize a fluctuated disparity surface. But the prediction shows that the smoothness regularization doesn't address this problem. By checking the training result in Figure 4.4, there is a big difference between the magnitude of L_{ap} and L_{smooth}.

This suggests the smoothness regularization is only valid in the early training stage, but becomes less effective later, as the magnitude difference between L_{ap} and L_{smooth} are too big.



1) Appearance loss L_{ap} 2) Smooth loss L_{smooth}

Figure 4.4.: Comparison between L_{ap} & L_{smooth}

We select 700 images in Gopro dataset to evaluate the prediction performance numerically. The same indicators are used as we choose in the KITTI experiment. Compared with KITTI one, the result here is better, but still doesn't outperform the original relative result.

Table 4.2.: Numerical result on Gopro

	Abs Rel	Sq Rel	RMSE	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Gopro (absolute)	0.3064	4.0621	11.6787	0.4787	0.7573	0.8689

The L_{abs} converges at around 0.07 meter level. This means the difference between true translation (range from 0.5 to 1.1 meter) and its prediction is only 0.07 meter, which is quite small.

Note that the value in Table 4.2 are the median over all 700 prediction indicators, rather than the average. This is because, as mentioned in section 3.2.1, the generated ground truth has some errors. Its influence will be mitigated if the median is used.

4.3. Evaluation of generalization ability

We also evaluate the generalization ability of our framework, the result is shown in Table 4.3.

Table 4.3.: Generalization ability of the framework

	Abs Rel	Sq Rel	RMSE	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Trained on Gopro Test on KITTI	0.3765	4.8738	10.5987	0.3512	0.6066	0.7559
Trained on KITTI Test on Gopro	0.5922	8.3166	14.3095	0.2424	0.5062	0.7587

It can be seen that the model trained on Gopro has a much better performance than the model trained on KITTI. The numerical accuracy tested on KITTI of the former model(trained on Gopro), is even better than performance of the latter model(trained on KITTI) on the KITTI test split.

For one hand this indicates that our framework have a good generalization ability. On the other hand, this also indicates the model trained on KITTI dataset only achieved a sub-optimum, as we analyzed before in section 4.1.

5. Conclusion & Discussion

In this project we exploit the single-view depth prediction based on view synthesis. Two network are jointly trained in our model, where Pose-net is supervised by the true translation, while Depth-net is self-supervised by the photometric error from the synthesized view. With introducing an absolute loss term, the network is able to output the absolute depth value. Although the prediction performance is inferior to the relative one, our framework shows that it is possible to eliminate the scale-ambiguity of depth prediction from *view synthesis* based method.

We also format our own Gopro dataset, covering various scenes and illumination conditions. We generate the depth ground truth for the test images, and evaluate our model on both KITTI and Gopro dataset.

From the experiment result, it's clear that the introduce of absolute loss term can upgrade the network from relative to absolute. It works better on Gopro dataset than KITTI, but both don't achieve the state-of-art prediction accuracy. Future improvement is possible through trying different parameters, like changing the magnitude of α_{smooth} or α_{abs}. We can also modify the network architecture. For example the Resnet50[21] can be used to replace the encoder part of Depth-net, as what Godard *et al.*[5] do in their work. The result shows that Resnet50 achieves a better prediction performance.

One remaining issue is that the depth prediction generally fails in texture-less area, like the image of expressway. As explained before, the failure is because the penalization for inaccurate depth from homogeneous part is low. This is one significant drawback of depth estimation whose supervision signal comes from the view synthesis(or generally, image reconstruction).

To address this problem some other constraints are required. In terms of the smoothness of depth, there is a huge difference between the appearance loss and smooth loss, as shown in Figure 4.4. We can employ a more strict smoothness regularization, like the one used in the work of Godard *et al.*[5] where the weight of disparity gradient is inversely proportional to the image gradient. The idea is that the depth discontinuities often occur at area where the image gradient is large.

It would be also interesting to utilize some semantic information. For example, we can identify which pixel can be labelled as ground, and then enforce the continuity of the homogeneous area on the predicted disparity map.

Another limitation is the premise to use view synthesis. According to [4], we implicitly assume the followings when use the photometric error as supervisory signal:

1. the scene is static without moving objects;
2. there is no occlusion/disocclusion between the target view and the source view;
3. the surface is Lambertian so that the photo-consistency error is meaningful.

However in some cases these assumption is not true. For example, there are some pedestrians walking around, and the nearby vehicles are also moving.

To address this problem, Zhou *et al.*[4] propose an *explainability* network to predict a pixel-wise mask for the target view. The network is trained to determine the dynamic object or occlusion/disocclusion area on the image. When computing the appearance loss these areas will be excluded.

However, this network doesn't improve the performance of depth prediction a lot. This is probably because the main error still comes from inaccurate depth in homogeneous area. After fixing the main problem, the improvement by adding this explainability network may become conspicuous.

Bibliography

[1] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374.

[2] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5162–5170.

[3] F. Liu, C. Shen, G. Lin, and I. D. Reid, "Learning depth from single monocular images using deep convolutional neural fields." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2024–2039, 2016.

[4] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, vol. 2, no. 6, 2017, p. 7.

[5] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, vol. 2, no. 6, 2017, p. 7.

[6] Y. Zou, Z. Luo, and J.-B. Huang, "Df-net: Unsupervised joint learning of depth and flow using cross-task consistency," in *European Conference on Computer Vision*. Springer, 2018, pp. 38–55.

[7] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," in *Advances in neural information processing systems*, 2006, pp. 1161–1168.

[8] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1253–1260.

[9] B. Li, C. Shen, Y. Dai, A. Van Den Hengel, and M. He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1119–1127.

[10] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 239–248.

[11] Y. Cao, Z. Wu, and C. Shen, "Estimating depth from monocular images as classification using deep fully convolutional residual networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 11, pp. 3174–3182, 2018.

[12] A. C. Kumar, S. M. Bhandarkar, and P. Mukta, "Depthnet: A recurrent neural network architecture for monocular depth prediction," in *1st International Workshop on Deep Learning for Visual SLAM,(CVPR)*, vol. 2, 2018, p. 2.

[13] A. Atapour-Abarghouei and T. P. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 18, 2018, p. 1.

[14] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *European Conference on Computer Vision*. Springer, 2016, pp. 740–756.

[15] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, "Sfm-net: Learning of structure and motion from video," *arXiv preprint arXiv:1704.07804*, 2017.

[16] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia, "Unsupervised learning of geometry with edge-aware depth-normal consistency," *arXiv preprint arXiv:1711.03665*, 2017.

[17] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5667–5675.

[18] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048.

[19] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3354–3361.

[20] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

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

[22] L. Ladicky, J. Shi, and M. Pollefeys, "Pulling things out of perspective," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 89–96.

A. Performance Indicator

In Chapter 4 6 numerical indicators are used to evaluate the depth prediction performance. These indicators are formulated as following. We use D_{pred} and D_{gt} to denote the ground truth and prediction of depth map. All computations are pixel-wise.

Abs Rel: short for *absolute relative error*, formulated as $\frac{1}{N}\sum |D_{pred} - D_{gt}| \ / \ D_{gt}$.

Sq Rel: short for *squared relative error*, formulated as $\frac{1}{N}\sum (D_{pred} - D_{gt})^2 \ / \ D_{gt}$.

RMSE: short for *root mean squared error*, formulated as $\sqrt{\frac{1}{N}\sum (D_{pred} - D_{gt})^2}$.

$\boldsymbol{\delta}$ refers to the relative difference between D_{pred} and D_{gt}. This indicator can be formulated as $\delta = max(D_{pred}/D_{gt}, D_{gt}/D_{pred})$. So the value in table for columns of $\delta < 1.25$, $\delta < 1.25^2$ and $\delta < 1.25^3$ are actually the percentage of δ whose magnitude is below 1.25, 1.25^2 and 1.25^3 respectively.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

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

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

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

Title of work (in block letters):

Semi-supervised Learning of Depth from Video

Authored by (in block letters):

For papers written by groups the names of all authors are required.

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

With my signature I confirm that

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

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

Place, date	**Signature(s)**
10.1.2018	Dai Qi

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