A Study on the Paper: Unsupervised Learning of Depth and Ego-Motion from Video

Burak Cem Balcı

Ege Dinçer

Dept. of Computer Engineering

Koc University

Istanbul, Turkey

I. Introduction

Understanding of 3D scenes remains to be an unsolved problem as computers fail to comprehend the properties of the 3D world by inspecting 2D samples of this world. While this task is very trivial for human beings, the perception of depth retrieved from a 2D image is not sufficient for computers to interpret it in the 3D world. In 2017 Zhou et al. proposed an unsupervised learning method to generate depth maps employing a single image as input [1]. This study aimed to imitate the training process of the original implementation by using the PyTorch framework. We achieved slightly worse results than the original implementation. In this report, we will first talk about the motivation behind Zhou et Al.'s and our work. Then we will discuss the methods employed in this work. Finally, we will demonstrate our implementation and interpret our results by comparing them with the results obtained by using the original implementation.

II. Motivation

A. Author's Motivation

Estimating the depth of a 2D image plays a crucial role in many domains of technology, especially for autonomous driving, unmanned aerial vehicles, and robotics. While this technology can be used for augmented reality, collision detection, virtual reality systems, Zhou et al. confirms that the traditional approaches failed to successfully achieve this goal. Employing supervised deep learning is a challenge for this problem as the number of labeled data is very limited. Moreover, the data labeling task is not sustainable for wide-ranged datasets.

To solve the mentioned problems feasibly Zhou et Al. proposed an unsupervised - or self-supervised- approach using deep learning architectures. Rather than employing pre-defined labels, this approach generates ground truths using epipolar geometry and easily accessed data such as camera intrinsics.

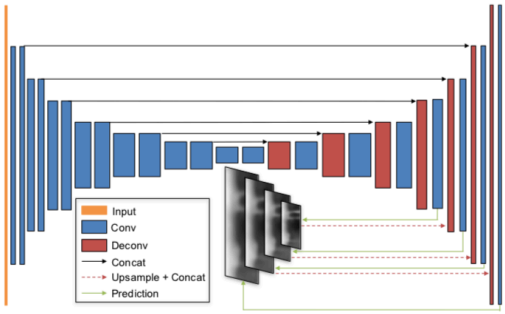
B. Our Motivation

This course was an ideal introduction to both deep learning and computer vision. As undergraduate students that are willing to pursue an academic career, we tried to find a topic that involves both computer vision and deep learning, which is considered a hot research area. With the supervision of Fatma Güney, we decided to work on this paper. The lecture introduced the PyTorch framework and this paper was originally implemented in Tensorflow 1.0, to have a better understanding of computer vision and novel frameworks used in deep learning re-implementing the paper using PyTorch was a suitable idea.

III. Approach

This paper is trying to generate a depth map of a frame, as mentioned previously. This work also provides a pose estimation network along with an explainability mask generator as a side outcome. These networks are trained simultaneously with a photometric loss function. This section will provide information about the networks are constructed and trained. We will go through the disparity, pose & explainability networks, related geometry knowledge, and loss functions.

A. Disparity Network

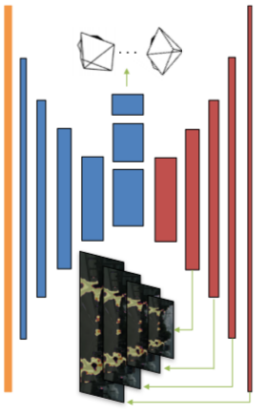


(a) Single-view depth network

The depth network is responsible for single image depth generation. It takes a frame at time t and it generates the depth map of that single image.

The network has the same architecture with the network proposed by Mayer et al in the paper "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation."[2]. It is called the DispNet model. It consists of 21 convolution and 7 deconvolution layers. It utilizes ReLu activation function in convolution layers and an activation function formulated as $1/(\alpha*\text{sigmoid}(x)+\beta)$ the hyper-parameters alpha and beta are set to 10 and 0.1 respectively. The layers of the network share a kernel size of 3 except for the first four convolutional layers. The first for convolutional layers use kernel sizes of 7,7,5 and 5 respectively. Finally, the first convolution layer has 32 output channels.

B. Pose Network



(b) Pose/explainability network

The pose network is responsible for generating the relative transformation matrices with respect to the image captured at time t. This network takes a sequence of images captured at time t-1, t and t+1. Although originally this network was meant to take a sequence of 3 images, the length of the sequence remains to be a hyper-parameter. Consequently, the output channel size of this network can be considered as 6 * the length of the sequence -1. Compared to the DispNet, the Pose network is shallower. It has 7 convolutional layers with a kernel size of 3 except the first two convolutional layers and the last two prediction layers. They have kernel sizes of 7,5,5,7 respectively. Similar to the DispNet architecture, it uses ReLu activation function, but it uses a linear activation function for the last layer.

C. Explainability Mask

The explainability network is employed to mask out vanishing, moving or occluded objects through the frames. The hold off related to these cases will be mentioned in the limitations section. Although the explainability mask may seem like a separate network, it's working within the pose network.

The explainability mask shares the first 5 convolutional layers of the Pose network. In addition to these convolutional layers, it features 5 deconvolutional layers. Similar to the Pose network, this network employs the ReLu activation functions despite the prediction layers, which utilize linear activation function.

D. Geometry Background

The paper states that, one can obtain the projected coordinates of a point on an image I_t by using the approximation.

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

This approximation states that a point p_t can be projected to an image source named I_s by using the camera intrinsics, relative pose, and depth map of the image. After projecting p_t to I_s as p_s, by using the bi-linear interpolation the value of warped image $\hat{I}_s$ is obtained. The paper construct its photo-metric loss function based on the assertion $I_t \sim \hat{I}_s$.

E. Loss Functions

The paper uses several loss functions. The first and main loss function is the photo-metric loss function introduced in section III-D. Both pose and disparity networks back-propagate according to this photometric loss function.

$$\mathcal{L}_{vs} = \sum_s \sum_p |I_t(p) - \hat{I}_s(p)|$$

According to the assertion introduced in section III-D, the loss should be 0 for perfect predictions of $T_{t\to s}$ and D_t. Back-propagating according to this loss function pushes both disparity and pose networks to approximate the $\hat{I}_s$ better and better.

The second loss function used is the smoothness loss. This loss is computed by minimizing L1 norm of the second order derivatives of the generated depth map. The loss is computed in order to overcome the gradient locality problem by allowing gradients to be directly extracted from bigger regions.

The explainability mask gives an output for each pixel that shows the confidence of the network about allowing the pixel to contribute to the loss function. When this term is added to the photometric loss the new loss becomes as shown below.

$$\mathcal{L}_{vs} = \sum_{<I_1,\dots,I_N>\in S} \sum_p \hat{E}_s(p)|I_t(p) - \hat{I}_s(p)| \,.$$

In order to avoid the obvious answer minimizing the loss function which is $E_s=0$ a regularization term consisting of E_s is added to the final loss function. Although the paper uses several loss functions the update is made over a final loss function which is the combination of the three functions mentioned.

$$\mathcal{L}_{final} = \sum_l \mathcal{L}^l_{vs} + \lambda_s \mathcal{L}^l_{smooth} + \lambda_e \sum_s \mathcal{L}_{reg}(\hat{E}^l_s)$$

IV. The Study Made

We started with following the lectures by Mubarak Shah concerning the topics structure from motion, stereo, camera model, fundamental matrix. After learning the basic geometry necessary for computer vision, we started imitating the original code using PyTorch. We created an additional Params class to keep parameters in a modular way. We implemented the training and network architecture parts however we used already implemented data loaders and some geometric functions. We used a portion of Kitti dataset during our training, as we were not able to train it with the full sized dataset in order to compare these to implementations we

trained both(original an our) of the models with the small portion of the dataset. Which is around 20GB.

V. LIMITATIONS

Initially, the paper assumes that the camera intrinsics are given so random videos or images cannot be fed to networks. Although the network is able to construct a depth map of the image it is not able to fully reconstruct the scene with 3D volumetric representations. One of the most important hold backs of this paper is the lack of understanding over scene dynamics and occlusions. The explainability mask is generated in order to make these segments drop form loss calculations. Even though it is not mentioned in the paper, during his conference presentation, the author tells that the network is not able to handle situations in which the front is wide open or there is a car in the middle. He suggests that this can be handled by increasing data amount and he says that the bias in the data might be causing this.

In our perspective we had difficulties handling such large datasets. Downloading ,storing and processing steps were not feasible. We solved this problem by only considering a small chunk of data. Colab was not able to meet our needs through the process due to crashes.

VI. CONCLUSION

Since we trained the original and our model on a small chunk of data, we were not able to achieve the error rates provided by the original paper. However, we were able to compare the loss values and error values over a very small scale. By considering the data used for training the difference between the numbers given in the paper and our results can be explained. We hold consistent results with the original model trained with small chunk of data. For the comparison we provide the results below.

abs_diff,	abs_rel,	sq_rel,	rms,	log_rms,	abs_log,	a1,	a2,	a3
5.7287,	0.3031,	4.0766,	10.3427,	0.4110,	0.2912,	0.5529,	0.7638,	0.8610

The above figure shows the results we have gathered through our calculation. A1, a2, a3, refers to the accuracy metrics introduced in the paper as $\delta<1.25$, $\delta<1.25^2$, and $\delta<1.25^3$ respectively.

The below figure shows the results of the original implementation

Abs Rel	Sq Rel	RMSE	RMSE(log)	Acc.1	Acc.2	Acc.3
0.183	1.595	6.709	0.270	0.734	0.902	0.959

Comparatively, our results have deviated because as mentioned earlier we were not able to train with the whole data-set, instead we have trained our data with a small-chunk of Kitti dataset (20 GB) of the whole data (Over 300 GB), which had caused the architecture to not extract the optimal parameters for the network . We have evaluated our implementation of the architecture with the data of 2011_09_26_drive_0009, and 2011_09_26_drive_0013.

References

[1]Zhou, Tinghui, et al. “Unsupervised Learning of Depth and Ego-Motion from Video.” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, doi:10.1109/cvpr.2017.700.

[2]Mayer, Nikolaus, et al. “A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation.” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, doi:10.1109/cvpr.2016.438.