# Learning the Structure from Motion, An Unsupervised Approach


Abhishek Kathpal
M. Eng Robotics
University of Maryland
College Park, Maryland 20740
Email: akathpal@umd.edu

Darshan Shah
M. Eng Robotics
University of Maryland
College Park, Maryland 20740
Email: dshah003@umd.edu

Mayank Pathak
M. Eng Robotics
University of Maryland
College Park, Maryland 20740
Email: pathak10@umd.edu


## I. Introduction

Until now, we have worked towards creating structure from motion using traditional methods which involved computing camera pose and scene depth from multiple images of a given scene. From the results it was apparent that the results created sparse output even after intense computation. There has been a lot of work done recently towards solving the structure from motion problem using Deep learning methods.

In this project we explore one such paper *Unsupervised Learning of Depth and Ego-Motion from Video* [6] and try to improvise the network with the goal of improving the accuracy more than that cited in the original paper. [6].

While studying the paper and exploring various possibilities, we tried to improve the network by several techniques. Few of which worked and few did not. In this report, we shall be addressing each of these things and finally compare our results with the ground truth and the results of the original paper.

## II. The SfM-Learner Network

The SfMLearner Network as described in [6] tries to predict the likely camera motion (Ego-motion) and scene structure and trains itself without supervision (Labelled data) by minimizing the loss function so as to get the predictions as close as possible to the ground truth. The network can be trained using sequence of images with no labeling or camera motion information.

The network consists of two different networks, each dedicated to training Depth and Pose respectively. The Depth network is realized using DispNet [1] which is an Encoder - Decoder based architecture with ReLU as activation function after each convolution network and sigmoid as prediction layer. The Depth Network takes one image frame as input and generates a depth map as output. On the other hand, Pose Network takes target view as an input which is concatenated with source views. The network synthesizes target image from multiple source images and outputs the relative camera poses. The output of both the networks are then used to inverse warp the source views to reconstruct target views and the photometric reconstruction loss is used for training the CNNs. Figure 1 shows the block diagram of SfMLearner network architecture.

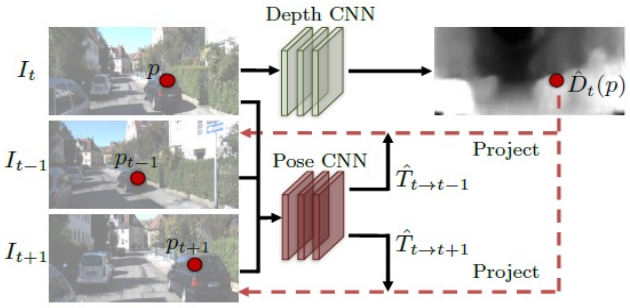



Fig. 1: Network Architecture of SfMLearner

The SfMLearner makes certain assumptions about the video feed. 1. Scenes are mostly rigid i.e. scene appearance across different frames is dominated by the camera motion. 2. There are no occlusion/dis-occlusions in the scene. 3. The surfaces in the scene are considered to be non-Lambertian surfaces. To improve the robustness of the learning pipeline to these factors, an additional network named 'explainability prediction network' is trained that outputs a per-pixel soft mask $E_s$ for each target source pair. Based on this predicted $E_s$ the view synthesis objective is weighted correspondingly.

This in a nutshell, summarizes the structure of the SfM-Learner network. As cited by [6] and verified by running the code on our systems, the network tends to converge after about 180K iterations. For this project, we restrict ourselves with training and testing on KITTI dataset. Table **??** tabulates the results obtained by the original SfM-learner and our modification of SfMLearner.

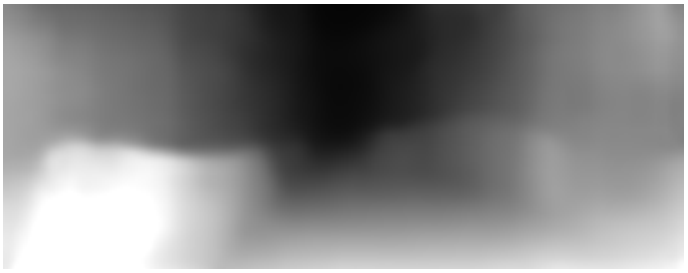
In the following sections, we shall discuss some of the modifications we've made in an attempt to improve the prediction of the Network.

## III. Network Improvisations

### A. Depth Estimation from Multiple views

The SfM-Learner takes just a single target view as input to compute depth map from the network and outputs a depth map. Instead, we implemented the network to take both target and source views as input. Doing so, provided a sharper depth map image and the network also tends to converge with lesser

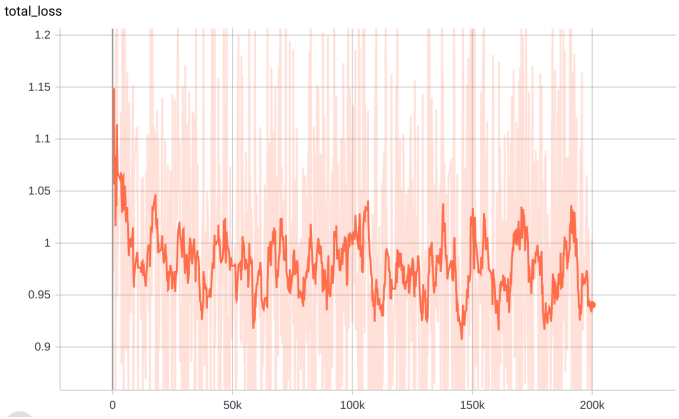
(a) Reference Image



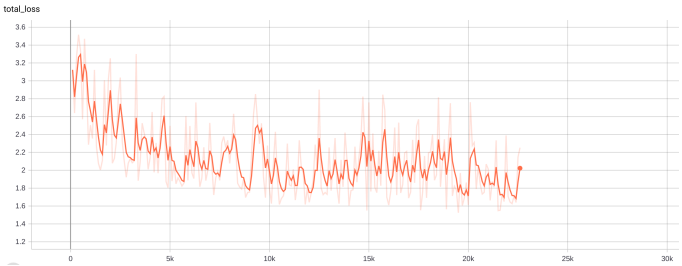
(b) Depth prediction of SfMLearner

(c) Depth prediction of modified method

Fig. 2: Depth output comparision between original, our output and ground truth




(a) SfMLearner




(b) Our output

Fig. 3: Total loss comparison

iterations than the single view implementation. The multiple frames given as input are nothing but the image sequences of the continuous video stream. For every target $t$ image, we also input successive frames $t+1$ and $t-1$ this accounts for and eliminates noises induced in depth map due to various factors. The multiple views leverages the relationship between pixels over multiple views to calculate depth (instead of relying on learned semantics) and help reinforcing the depth estimates and provides a depth prior for better depth map creation. Figure **??** shows the depth map output of single view and multi-view inputs.

### *B. Structural Similarity Loss*

SfM-Learner relies on minimization of photometric loss for training it's network. While the photometric loss gives pretty good results, it makes certain assumptions such as scenes requiring to have a constant brightness, luminosity and so forth. This constraint does not necessarily hold true in all cases. To address this problem we explore an different kind of metric Structural similarity Index (SSIM) as described in [4]. The Luminance of surface of an object is a product of illumination and reflectance, but the structure of an object are independent of illumination. SSIM provides a robust metric for measuring perpetual differences between two images by considering the 3 factors of luminance, contrast and structure. We added the SSIM loss term to the final loss. Since Structural similarity index is usually maximized (with the maximum value being 1), we minimize the below function

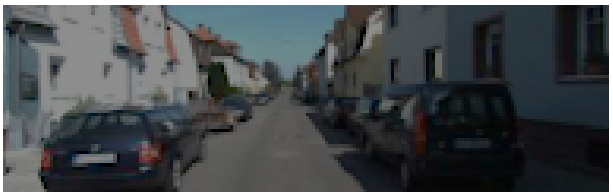
$$L_{ssim} = \sum_{s} \frac{1 - SSIM(I_t, I_s)}{2} \tag{1}$$

### *C. Adaptive learning Rate*

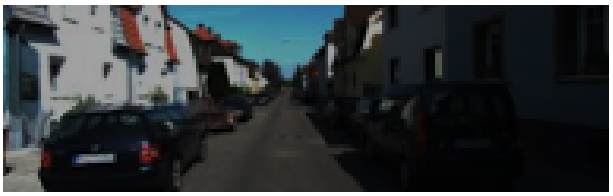
The Original SfMlearner used a constant learning rate of 0.0002. With a batch size of 4, this took about 200 epochs before the network converges. Since the loss function took a long time before the values could drop to considerable amount. Hence we implemented an adaptive learning rate, which starts with a high value of 0.002 but gradually reduces by 20% at every step of 10,000 iterations (15 epochs). Figure 3 shows a plot comparison of loss functions of original paper (as implemented on our computer) and the final loss curve obtained from our improvised hyper-parameter. These plots are obtained using Tensorboard.

### *D. Batch Normalization*

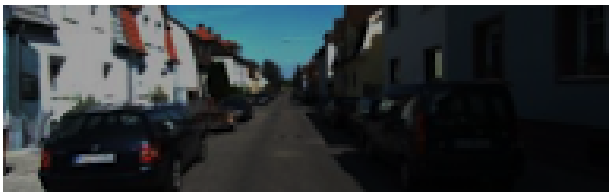
With the goal of further regularizing the network, we also implemented batch normalization after every 4 conv layers. Unfortunately, it had a negative impact on the model and



(a) Color fading



(b) brightness change



(c) saturation change

Fig. 4: Sample of how input data is augmented by changing brightness, coloring, saturation and shifting gamma

increased the overall loss. Hence, we decided to discard batch normalization from our implementation. This was probably because of over regularization.

### *E. Data Augmentation*

Data Augmentation is an important factor that can improve results significantly. With this, possible scenarios that can come across during real-time testing like varying brightness, different orientations and scales are taken into account. SfM-Learner already include some data augmentation by randomly scaling and cropping the input data. In addition to that, we included random shifting of the gamma values (making regions darker or lighter), randomly changing brightness and color of the images. These lead to improvements in loss.

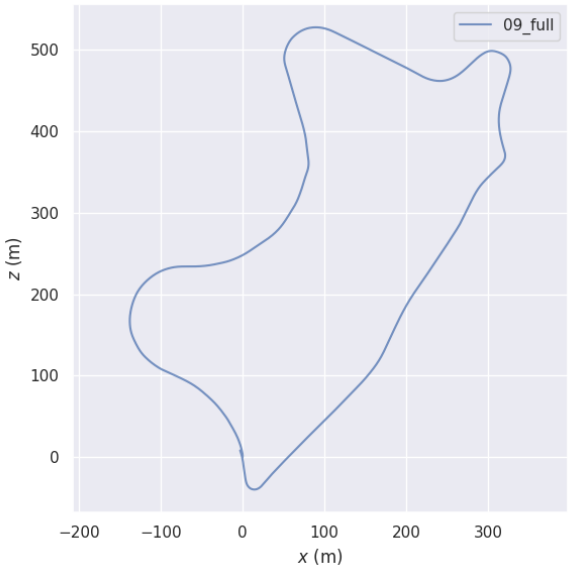
## IV. Results

After implementing the Network changes mentioned in the previous sections, we train our network for 200,000 iterations with a Mini-batch size of 16 on a Nvidia Tesla P100 GPU. The average time per iteration came out to be about 33 milliseconds per iteration. The SfM learner took about 22 milliseconds but with a mini batch size of 4.

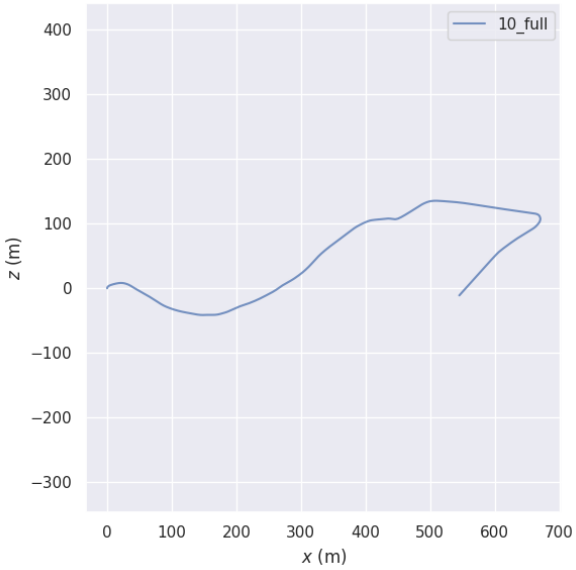
Figures 2 through 7 demonstrates the result comparison between original approach and our approach.

Figure 2 shows the difference between the depth map output between the depths generated by SfMLearner and our upgraded version. The Figure 7 shows various metric graphs of loss functions evolving over the training period

The Loss metrics are summarized in table III.




(a) ground truth trajectory for seq. 09




(b) ground truth trajectory for seq. 10

Fig. 5: Odometry output from data

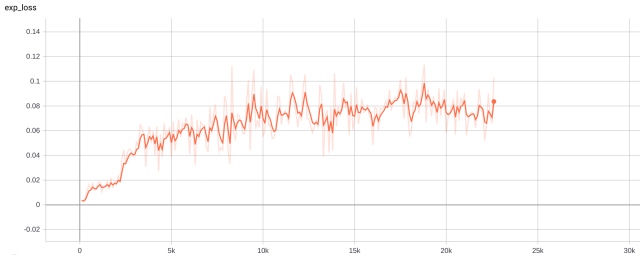
| Model | ATE mean | std |
|---|---|---|
| Original | 0.0215 | 0.0131 |
| Modified | 0.0142 | 0.0089 |

TABLE I: Pose error comparison between original approach and our Implementation for seq 9

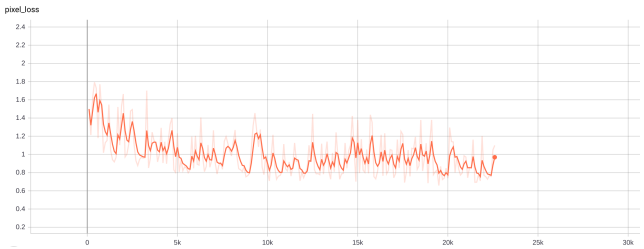
| Model | ATE mean | std |
|---|---|---|
| Original | 0.0091 | 0.0070 |
| Modified | 0.0153 | 0.0118 |

TABLE II: Pose error comparison between original approach and our Implementation for seq 10

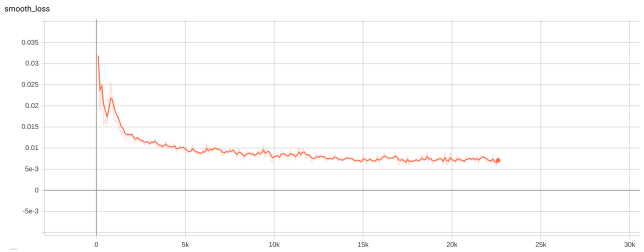
Fig. 6: Some input(left) and Output(right) images from our implementation.



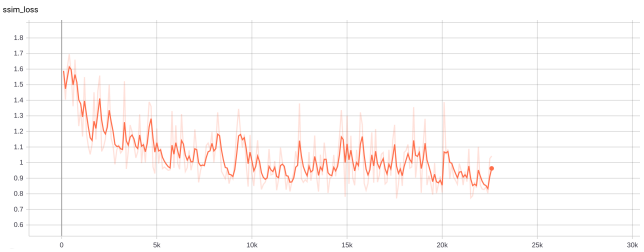

(a) explanability loss



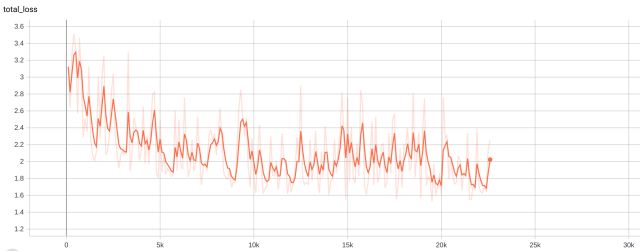

(b) pixel loss




(c) smooth loss




(d) SSIM loss




(e) Total loss

Fig. 7: Error progression for our approach

| SfM Learner Model | abs_rel | sq_rel | rms | log_rms | d1_all | a1 | a2 | a3 |
|---|---|---|---|---|---|---|---|---|
| As Cited | 0.2003 | 1.8324 | 6.8304 | 0.2810 | 0.0000 | 0.7122 | 0.8944 | 0.9555 |
| Trained by us | 0.2655 | 2.3823 | 8.0507 | 0.3484 | 0.0000 | 0.5608 | 0.8413 | 0.9265 |
| Our Implementation | 0.4402 | 4.8310 | 12.4181 | 0.5890 | 0.0000 | 0.3008 | 0.5602 | 0.7706 |

TABLE III: Error comparison between original approach and our output

## V. Discussion and Possible Modifications

There are many possible modifications that we thought of which might lead in improvements of results, but due to limited time we were not able to implement them. The following possible modifications are listed below-

1. Instead of using a simple CNN for depth estimation methods, we can make use of Long Short-Term Memory (LSTM) units with convolutional layers to effectively utilize multiple previous frames in each estimated depth maps. This will lead to improvements in computation of depth maps.The inspiration for this idea is taken from this paper [3], it takes into both SFMLearner and Geonet Architectures into account. [5]

2. SFM does not take into epipolar constraints into account. We can incorporate epipolar constraints when computing loss. This constraints require us to compute the Essential matrix which can be done using nister 5-point algorithm. The Essential Matrix contains information about the relative poses between the views. Similar concept is explained in [2].

## References


[1] Nikolaus Mayer et al. "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation". In: *CoRR* abs/1512.02134 (2015). arXiv: 1512.02134. URL: http://arxiv.org/abs/1512.02134.

[2] Vignesh Prasad, Dipanjan Das, and Brojeshwar Bhowmick. "Epipolar Geometry based Learning of Multi-view Depth and Ego-Motion from Monocular Sequences". In: *arXiv preprint arXiv:1812.11922* (2018).

[3] Rui Wang, Jan-Michael Frahm, and Stephen M Pizer. "Recurrent neural network for learning densedepth and ego-motion from video". In: *arXiv preprint arXiv:1805.06558* (2018).

[4] Zhou Wang et al. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE TRANSACTIONS ON IMAGE PROCESSING* 13.4 (2004), pp. 600–612.

[5] Zhichao Yin and Jianping Shi. "Geonet: Unsupervised learning of dense depth, optical flow and camera pose". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1983–1992.

[6] Tinghui Zhou et al. "Unsupervised Learning of Depth and Ego-Motion from Video". In: *CVPR*. 2017.