

VISUAL ODOMETRY AND DEPTH MAPS ESTIMATION

Self-driving cars



Omar Khaled	5562
Mariam Matar	5653
Mariam Fekry	5614
Lina Hazem	5613

TABLE OF CONTENTS

INTRODUCTION.....	3
--------------------------	----------

PAPER'S EXPERIMENTS	4
----------------------------------	----------

Introduction.....	4
Network's Architecture	5
Loss Functions	8
Paper Results.....	12
Paper's Ablation Study	13

OUR EXPERIMENTS.....	16
-----------------------------	-----------

Introduction.....	16
Environment Setup.....	16
Data Preparation	16
Running Our Experiment	17
Our Results	38
Observations	29
What Is Next?	29
Proposed Modifications.....	30

Introduction

In this report we summarize the ideas provided by the paper “Recurrent Neural Network for (Un-)supervised Learning of Monocular Video Visual Odometry and Depth” by Wang et al. -CVPR 2019-, and the results of our experiments with its architecture provided by the authors and the modifications that we want to do to the proposed architecture and its losses.

Recently deep learning-based, single-view depth estimation methods have shown highly promising results. However, such methods ignore one of the most important features for determining depth in the human vision system, which is motion.

The paper proposes a learning-based, multi-view dense depth map and odometry estimation method that uses Recurrent Neural Networks (RNN) and trains utilizing multi-view image reprojection and forward-backward flow-consistency losses.

The model can be trained in a supervised or even unsupervised mode. It is designed for depth and visual odometry estimation from video where the input frames are temporally correlated. However, it also generalizes to single-view depth estimation.

First, we are going to go through the ideas of the paper and the intuition behind it and then we are going to summarize the results of our experiments.

Paper's Experiments

Introduction

The power of the "Recurrent Neural Network for (Un-)supervised Learning of Monocular Video Visual Odometry and Depth" paper was in the **ConvLSTM** layers which they used to produce the depth maps by interleaving the **ConvLSTM** layers within the normal encoder-decoder with skip connections for image reconstruction and for this we quote the authors of the paper when they said : " Recently, convolutional neural networks (CNN) have begun to produce results of comparable quality to traditional geometric computer vision methods for depth estimation in measurable areas and achieve significantly more complete results for ambiguous areas through the learned priors. However, in contrast to traditional methods, most CNN methods treat depth estimation as a single view task and thus ignore the important temporal information in monocular or stereo videos. The underlying rationale of these single view depth estimation methods is the possibility of human depth perception from a single image. However, they neglect the fact that motion is actually more important for the human to infer distance." The authors say that they have improved upon existing deep single- and two-view stereo depth estimation methods by interleaving **ConvLSTM** units with the convolutional layers to effectively utilize multiple previous frames in each estimated depth maps. Since they utilize multiple views, the image reprojection constraint between multiple views can be incorporated into the loss, which shows significant improvements for both supervised and unsupervised depth and camera pose estimation.

NETWORK ARCHITECTURE

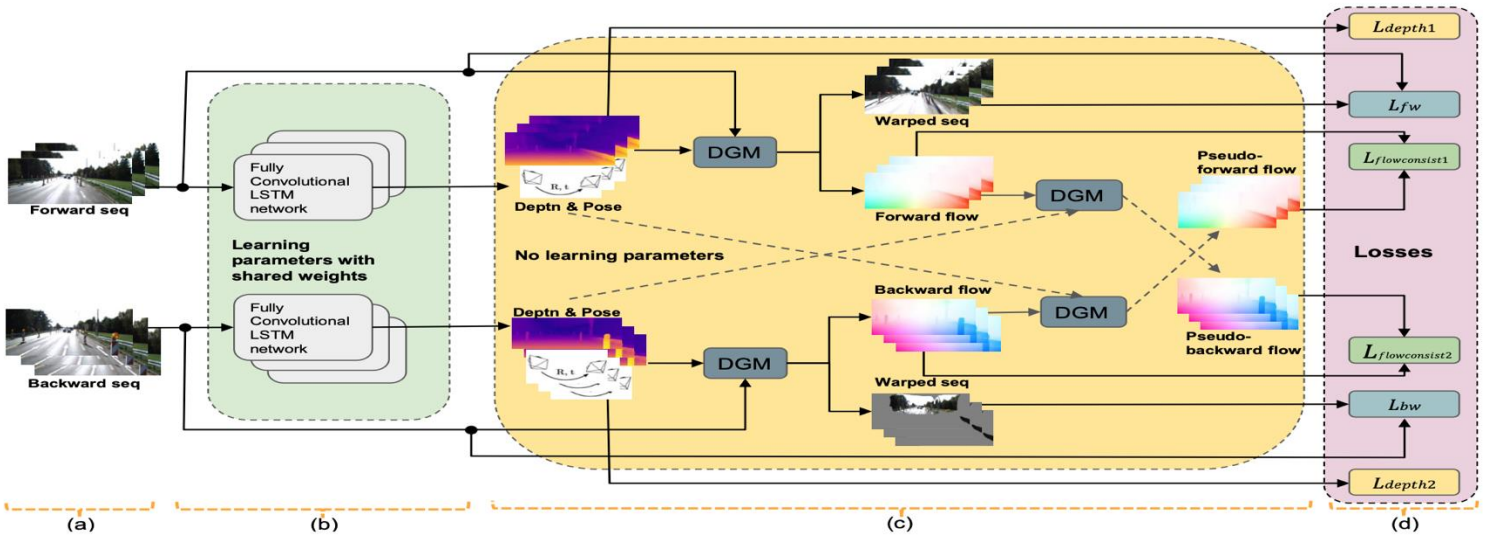


Figure 1: Training pipeline of the proposed RNN-based depth and visual odometry estimation network.

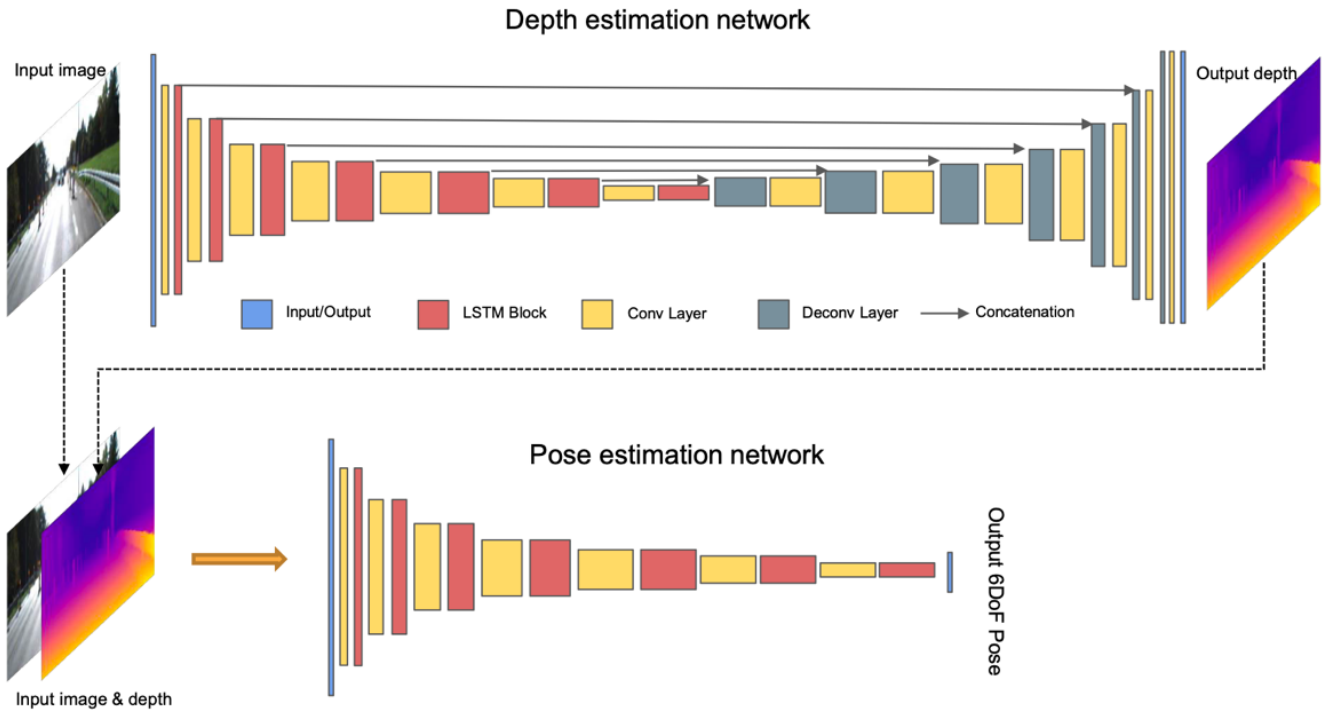


Figure 2: Depth estimation network

The architecture, shown in Figure 2, is made up of two networks, one for depth and one for visual odometry.

Type	Filters	Output size
Input		$128 \times 416 \times 3$
Conv+ConvLSTM	$32@3 \times 3 \times 3$	$64 \times 208 \times 32$
Conv+ConvLSTM	$64@3 \times 3 \times 32$	$32 \times 104 \times 64$
Conv+ConvLSTM	$128@3 \times 3 \times 64$	$16 \times 52 \times 128$
Conv+ConvLSTM	$256@3 \times 3 \times 128$	$8 \times 26 \times 256$
Conv+ConvLSTM	$256@3 \times 3 \times 256$	$4 \times 13 \times 256$
Conv+ConvLSTM	$256@3 \times 3 \times 256$	$2 \times 7 \times 256$
Conv+ConvLSTM	$512@3 \times 3 \times 256$	$1 \times 4 \times 512$
Deconv+Concat+Conv	$256@3 \times 3 \times 512$	$2 \times 7 \times 256$
Deconv+Concat+Conv	$128@3 \times 3 \times 256$	$4 \times 13 \times 128$
Deconv+Concat+Conv	$128@3 \times 3 \times 128$	$8 \times 26 \times 128$
Deconv+Concat+Conv	$128@3 \times 3 \times 128$	$16 \times 52 \times 128$
Deconv+Concat+Conv	$64@3 \times 3 \times 128$	$32 \times 104 \times 64$
Deconv+Concat+Conv	$32@3 \times 3 \times 64$	$64 \times 208 \times 32$
Deconv	$16@3 \times 3 \times 32$	$128 \times 416 \times 16$
Conv (output)	$1@3 \times 3 \times 16$	$128 \times 416 \times 1$

Table 1: Detailed depth estimation network architecture. Every convolution in the encoder uses **stride 2 for downsampling**. Before the output a **sigmoid** activation function is used to ensure the output is in range $[0, 1]$; **All the other convolutions and decovolutions are followed by batch norm and LeakyRELU activation.**

Type	Filters	Output size
Input		$128 \times 416 \times 4$
Conv+ConvLSTM	$32@3 \times 3 \times 3$	$64 \times 208 \times 32$
Conv+ConvLSTM	$64@3 \times 3 \times 32$	$32 \times 104 \times 64$
Conv+ConvLSTM	$128@3 \times 3 \times 64$	$16 \times 52 \times 128$
Conv+ConvLSTM	$256@3 \times 3 \times 128$	$8 \times 26 \times 256$
Conv+ConvLSTM	$256@3 \times 3 \times 256$	$4 \times 13 \times 256$
Conv+ConvLSTM	$256@3 \times 3 \times 256$	$2 \times 7 \times 256$
Conv+ConvLSTM	$512@3 \times 3 \times 256$	$1 \times 4 \times 512$
Conv (output)	$6@1 \times 1 \times 512$	$1 \times 1 \times 6$

Table 2: Detailed visual odometry network architecture. Every convolution (except for output layer) is followed by batch normalization and RELU as activation.

The depth estimation network uses a U-shaped network architecture like DispNet. The main innovation is to interleave recurrent units into the encoder which allows the network to leverage not only spatial but also temporal information in the depth estimation. The spatial-temporal features computed by the encoder are then fed into the decoder for accurate depth map reconstruction. The input to the depth estimation network is a single RGB frame I_t and the hidden states h^d_{t-1} from the previous time-step (h^d_{t-1} are initialized to be all zero for the first time-step). The hidden states are transmitted internally through the ConvLSTM units. The outputs of the depth estimation network are the depth map Z_t and the hidden states h^d_t for the current time-step.

The visual odometry network uses a VGG16 architecture with recurrent units interleaved. Table 2 shows the detailed network architecture. The input to the visual odometry network is the concatenation of I_t and Z_t together with the hidden states h_{t-1}^p from the previous time-step. The output is the relative 6DoF camera pose $P_{t \rightarrow t-1}$ between the current view and the immediately preceding view. The main differences between the visual odometry network and most current deep learning-based visual odometry methods are 1) At each time-step, instead of a stack of frames, The visual odometry network only takes the current image as input; the knowledge about previous frames is in the hidden layers. 2) The visual odometry network also takes the current depth estimation as input, which ensures a consistent scene scale between depth and camera pose (important for unsupervised depth estimation, where the scale is ambiguous). 3) The visual odometry network can run on a full video sequence while maintaining a single scene scale.

Loss Functions

- *Multi-view Reprojection Loss*

The learning of depth and visual odometry estimation can be formulated as an image reconstruction problem using a **differentiable geometric module (DGM)**. Thus, the DGM can be used to formulate an image reconstruction constraint between I_t and I_{t-1} using the estimated depth Z_t and camera pose $P_{t \rightarrow t-1}$. As shown in Figure 1(c), the output depth maps, and relative camera poses together with the input sequence are fed into a differentiable geometric module (DGM) that performs differentiable image warping of every previous view of the sub-sequence into the current view. Denote the input image sequence (shown in Figure 2(a)) as $\{I_t | t = 0 \dots N - 1\}$, the estimated

depth maps as $\{Z_t | t = 0 \dots N - 1\}$, and the camera poses as the transformation matrices from frame t to $t-1$: $\{P_{t \rightarrow t-1} | t = 0 \dots N - 1\}$.

The multi-view reprojection loss equation is

$$L_{fw} = \sum_{t=0}^{N-1} \sum_{i=0}^{t-1} \sum_{\Omega} \lambda_t^i \omega_t^i |I_t - \hat{I}_t^i| \quad (1)$$

where \hat{I}_t^i is the i^{th} view warped into t^{th} view, Ω is the image domain, ω_t^i is a binary mask indicating whether a pixel of I_t has a counterpart in I_i , and λ_t^i is a weighting term that decays exponentially based on $t - i$. Image pairs that are far away naturally suffer from larger reprojection error due to interpolation and moving foreground so we use λ_t^i to reduce the effect of such artifacts. ω_t^i and \hat{I}_t^i are obtained as

$$\omega_t^i, \hat{I}_t^i, F_{t \rightarrow i} = \phi(I_i, Z_t, P_{t \rightarrow i}, K) \quad (2)$$

where $F_{t \rightarrow i}$ is a dense flow field for 2D pixels from view t to view i , which is used to compute flow consistency. K is the camera intrinsic matrix. The pose change from view t to i , $P_{t \rightarrow i}$ can be obtained by a composite transformation as

$$P_{t \rightarrow i} = P_{i+1 \rightarrow i} \cdot \dots \cdot P_{t-1 \rightarrow t-2} \cdot P_{t \rightarrow t-1} \quad (3)$$

The function ϕ in Equation 2 warps image I_i into I_t using Z_t and $P_{t \rightarrow i}$. The function ϕ is a DGM, which performs a series of differentiable 2D-to-3D, 3D-to-2D projections, and bi-linear interpolation operations.

In the same way, we reverse the input image sequence and perform another pass of depth $\{Z_t | t = N - 1 \dots 0\}$ and camera pose $\{P_{t \rightarrow t+1} | t = N - 1 \dots 0\}$ estimation, obtaining the backward multi-view reprojection loss L_{bw} .

- *Forward-backward Flow Consistency Loss*

We first introduce a forward-backward consistency constraint on a single pair of frames and then generalize to a sequence. Let us denote a pair of consecutive frames as I_A and I_B , and their estimated depth maps and relative poses as Z_A , Z_B , $P_{A \rightarrow B}$, and $P_{B \rightarrow A}$. We can obtain a dense flow field $F_{A \rightarrow B}$ from frame I_A to I_B using Equation 2. Similarly, we can obtain $F_{B \rightarrow A}$ using Z_B , $P_{B \rightarrow A}$. Using $F_{B \rightarrow A}$ we can compute a pseudo-inverse flow $\hat{F}_{A \rightarrow B}$ (due to occlusion and interpolation) as

$$\omega_A^B, \hat{F}_{A \rightarrow B}, F_{A \rightarrow B} = \phi(-F_{B \rightarrow A}, Z_A, P_{A \rightarrow B}, K) \quad (4)$$

This is similar to Equation 2 except that we are interpolating $F_{A \rightarrow B}$ from $-F_{B \rightarrow A}$ instead of I_t from I_i . Therefore, we can formulate the flow consistency loss as

$$L_{flowconsit} = \omega_A^B \cdot |F_{A \rightarrow B} - \hat{F}_{A \rightarrow B}| + \omega_B^A \cdot |F_{B \rightarrow A} - \hat{F}_{B \rightarrow A}| \quad (5)$$

This is performed for every consecutive pair of frames in the input sequence. Unlike the multi-view reprojection loss we only compute flow-consistency on pairs of consecutive frames given the fact that the magnitude of the flow increases, for frame pairs that are far apart, leading to inaccurate pseudo-inverses due to interpolation.

- *Smoothness Loss*

An edge-aware smoothness constraint can be defined as

$$L_{smooth} = \sum_{t=0}^{N-1} \sum_{\Omega} |\nabla \xi_t| \cdot e^{-|\nabla I_t|} \quad (6)$$

- *Absolute depth loss*

All stated before can be used for unsupervised training. However, if there is groundtruth depth available, even sparsely, we can train a network to estimate depth at absolute scale by adding the absolute depth loss defined as

$$L_{depth} = \sum_{t=0}^{N-1} \sum_{\Omega} |\xi_t - \hat{\xi}_t| \quad (7)$$

In addition, the local smoothness loss in Equation 6 can be replaced by a gradient similarity to the groundtruth depth, which can be defined as

$$L_{smooth} = \sum_{t=0}^{N-1} \sum_{\Omega} |\nabla \xi_t - \nabla \hat{\xi}_t| \quad (8)$$

PAPER RESULTS

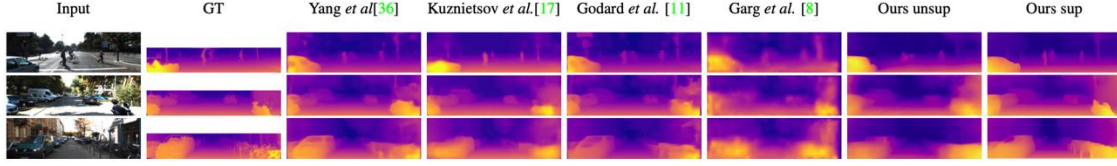


Figure 4: Visual comparison between the state-of-the-art methods. For visualization the groundtruth depth is interpolated. Our method captures more details in thin structures, such as the motorcycle and columns in the lower right corner of figure rows 2 and 3.

Methods	Dataset	Supervised		Error metric				Accuracy metric		
		depth	pose	RMSE	RMSE log	Abs Rel	Sq Rel	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Zhou et al. [42]	CS+K			6.709	0.270	0.183	1.595	0.734	0.902	0.959
Liu et al. [20]	K	✓		6.523	0.275	0.202	1.614	0.678	0.895	0.965
Eigen et al. [4]	K	✓		6.307	0.282	0.203	1.548	0.702	0.890	0.958
Yin et al. [38]	K			5.857	0.233	0.155	1.296	0.806	0.931	0.931
Zhan et al. [40]	K		✓	5.585	0.229	0.135	1.132	0.820	0.933	0.971
Zou et al. [44]	K			5.507	0.223	0.150	1.124	0.793	0.933	0.973
Godard et al. [11]	CS+K		✓	5.311	0.219	0.124	1.076	0.847	0.942	0.973
Atapour et al. [1]	K+S*	✓		4.726	0.194	0.110	0.929	0.923	0.967	0.984
Kuznetsov et al. [17]	K	✓	✓	4.621	0.189	0.113	0.741	0.875	0.964	0.988
Yang et al. [36]	K	✓		4.442	0.187	0.097	0.734	0.888	0.958	0.980
Fu et al. (ResNet) [7]	K	✓		2.727	0.120	0.072	0.307	0.932	0.984	0.994
Ours-unsup (multi-view)	K			2.320	0.153	0.112	0.418	0.882	0.974	0.992
Ours-sup (single-view)	K	✓		<u>1.949</u>	0.127	0.088	<u>0.245</u>	0.915	0.984	<u>0.996</u>
Ours-sup (multi-view)	K	✓		1.698	0.110	<u>0.077</u>	0.205	0.941	0.990	0.998

Table 3: Quantitative comparison of our network with other state-of-the-art CNN-based methods on KITTI [10] dataset using the Eigen Split [4]. *Ours sup* (Single-view) is the evaluation of single view depth estimation result. *Ours sup* (multi-view) is the result generated with the assistance of nine previous views. Even though our method is not restricted to a fixed number of frames per sequence during prediction or evaluation, we still use 10-frame sequence here for the consistency with the training. We discuss continuous estimation results in the ablation study Section 4.5. The bold numbers are results that rank first and the underlined results those that rank second. All results are capped at 80m depth.

Methods	Seq 09		Seq 10	
	$t_{err}(\%)$	$r_{err}(\text{deg/m})$	$t_{err}(\%)$	$r_{err}(\text{deg/m})$
ORB-SLAM [25]	15.30	0.003	3.68	0.005
GeoNet [38]	43.76	0.160	35.60	0.138
Zhou et al. [42]	17.84	0.068	37.91	0.178
Zhan et al. [39]	11.92	0.036	12.62	0.034
DeepVO et al. [34]	-	-	8.11	0.088
Our unsupervised	9.88	0.034	12.24	0.052
Our supervised	9.30	0.035	7.21	0.039

Table 4: Quantitative comparison of visual odometry results on the KITTI Odometry dataset. t_{err} is the percentage of average translational error and r_{err} is the average degree per meter rotational error.

PAPER'S ABLATION STUDY

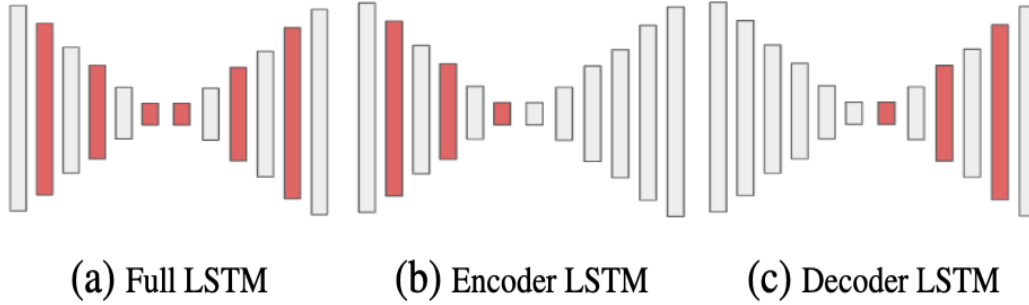


Figure 6: Three different architectures depend on the placements of recurrent units. (a) We put a convolutional LSTM after every convolution or deconvolution layer. (b) We only place convolutional LSTM in the encoder. (c) We only place convolutional LSTM in the decoder.

Method	RMSE	RMSE log	Abs Rel	Sq Rel
full LSTM	1.764	0.112	0.079	0.214
decoder LSTM	1.808	0.117	0.082	0.226
encoder LSTM	1.698	0.110	0.077	0.205

Table 5: Ablation study on network architectures. The evaluation data and protocol are the same as table 3.

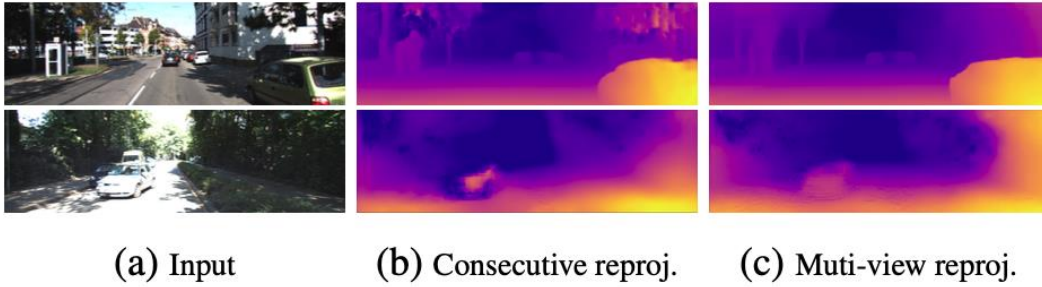


Figure 7: Visual examples between networks trained using consecutive image reprojection loss and those using multi-view reprojection loss. Results in the first row are from *ours-sup*, and results in the second row are from *ours-unsup*.

Method	RMSE	RMSE log	Abs Rel	Sq Rel
Ours-d	1.785	0.116	0.081	0.214
Ours-dc	1.759	0.113	0.079	0.215
Ours-mc	1.698	0.110	0.077	0.205
Ours-dc unsup	2.689	0.184	0.138	0.474
Ours-mc unsup	2.361	0.157	0.112	0.416

Table 6: Ablation study on multi-view reprojection and forward-backward flow consistency constraints. *d* stands for consecutive image reprojection. *m* stands for multi-view image reprojection. *c* stands for forward-backward flow consistency constraint. The first three rows are comparison between supervised training and the last two rows are unsupervised.

Window size	RMSE	RMSE log	Abs Rel	Sq Rel
1	1.949	0.127	0.088	0.245
3	1.707	0.110	0.077	0.206
5	1.699	0.110	0.077	0.205
10	1.698	0.110	0.077	0.205
20	1.711	0.117	0.077	0.208
Whole seq.	1.748	0.119	0.079	0.214

Table 7: Depth estimation with different time-window sizes.

2) the performance of the depth estimation is not increasing after 10 frames; 3) even though our network is trained on 10-frame based sub-sequences, it can succeed on an arbitrary length sequences.

Our Experiments

INTRODUCTION

We started setting up the environment, in which we faced a lot of obstacles with the frameworks and the data preparation phase after this we started training the model with the papers provided hyperparameters and on all the KITTI and KITTI depth dataset for 135000 steps (27 epochs).

And we provide our results after these 27 epochs which were less than the number of epochs stated in the paper which are a minimum of 30 or 40 but we ran out of time and didn't have enough resources to continue so we propose some modifications that can be done to the architecture and training phase in order to improve the paper results.

ENVIRONMENT SETUP

After a couple of unsuccessful trials, we settled on working on Google colab pro plus on which we faced a lot of obstacles due to the incompatibility of the environment with the model's code, which was written in python 3.6, tensorflow 1.12 and on CUDA 10.1 which was challenging to fix in the colab containerized environment but after lots of trials we managed to fix these issues, but it took us sometime.

DATASET PREPARATION

First, we downloaded the KITTI and KITTI depth data set which took a lot of time to download then we ran the preprocessing code provided by the paper authors which remove sequences with very low change between frames and prepare the images dimensionality and generate the forward optical flow of the images offline the dataset was splitted using the KITTI eigen split as stated by the authors of the paper.

RUNNING OUR EXPERIMENT

After overcoming all the obstacles that faced us, we started our training procedure using the same hyperparameters provided by the authors which are:

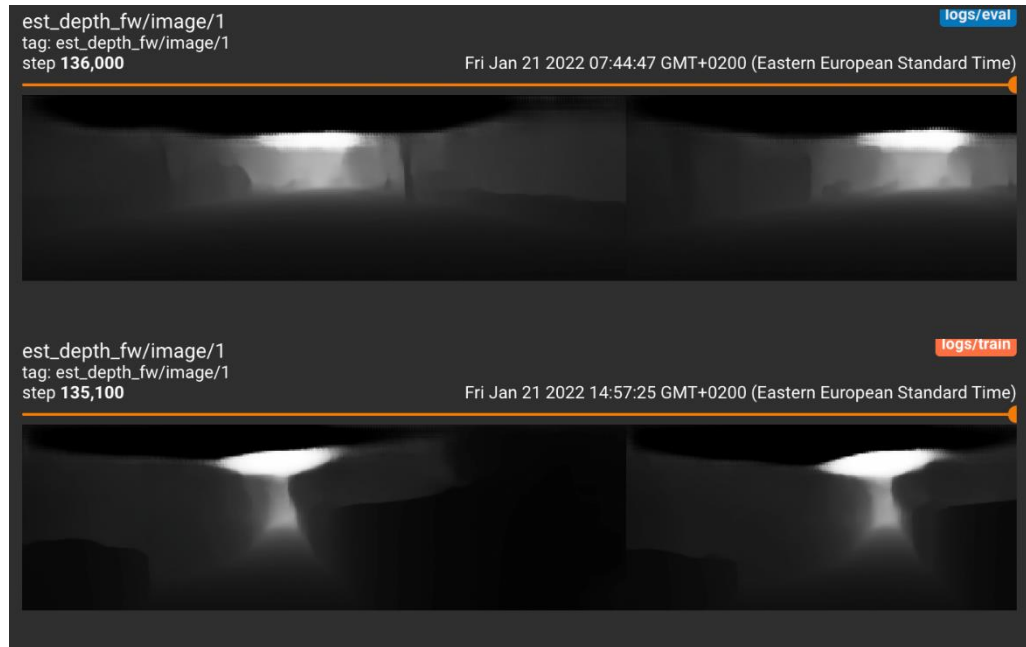
- Batch size: 3
- Image height: 128
- Image width: 416
- Number of views per input (frames): 10
- Learning Rate: 0.0001
- Adam optimizer with beta: 0.9

We trained for 135000 steps or 27 epochs 3 epochs each day as we were limited by the colab p100 GPU runtime of maximum 24 hours.

OUR RESULTS

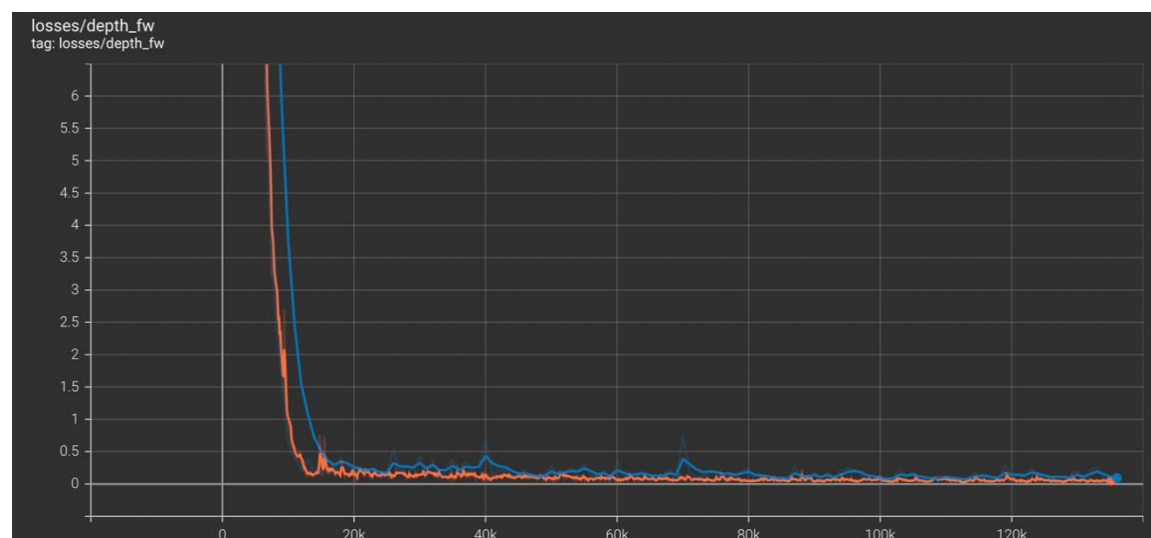
Depth Estimation

- *Estimated forward depth maps for different images*

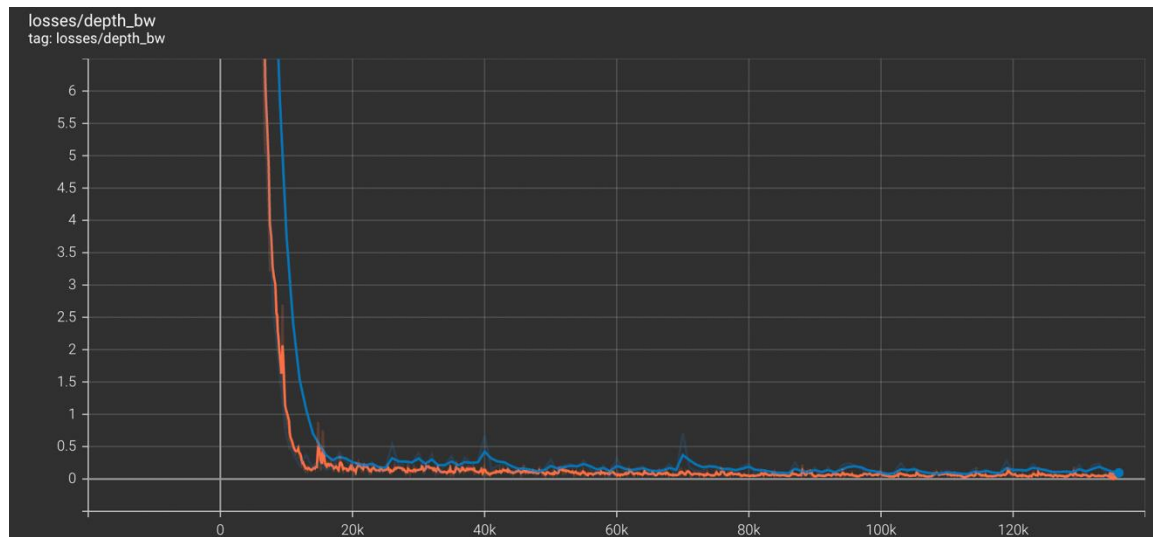




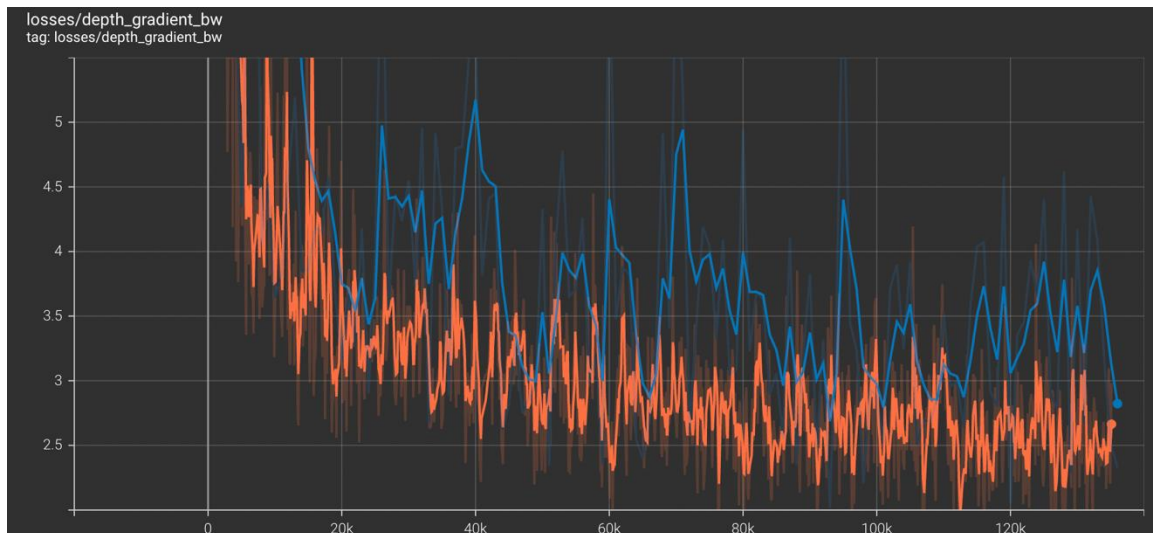
- *Forward depth estimation loss*



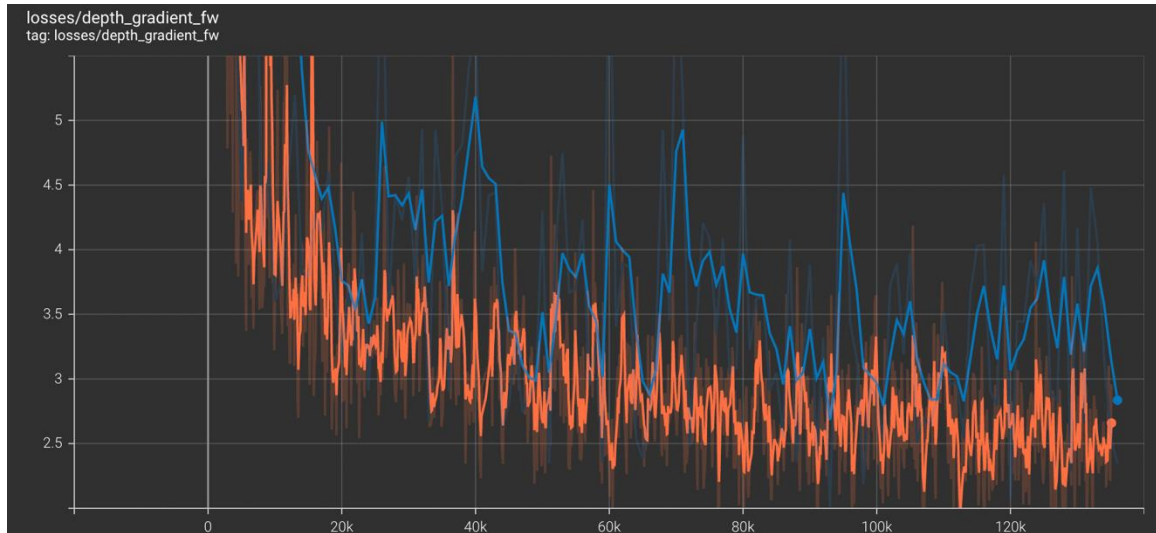
- *Backward depth estimation loss*



- *Backward depth gradient loss*



- *Forward depth gradient loss*



Forward-Backward Flow-Consistency

- *Projected images*



tower_0/proj_img_bw/image

logs/train

tag: tower_0/proj_img_bw/image

step 135,100Fri Jan 21 2022 14:57:25 GMT+0200 (Eastern European Standard Time)



tower_0/proj_img_fw/image

logs/eval

tag: tower_0/proj_img_fw/image

step 136,000 Fri Jan 21 2022 07:44:47 GMT+0200 (Eastern European Standard Time)



tower_0/proj_img_fw/image

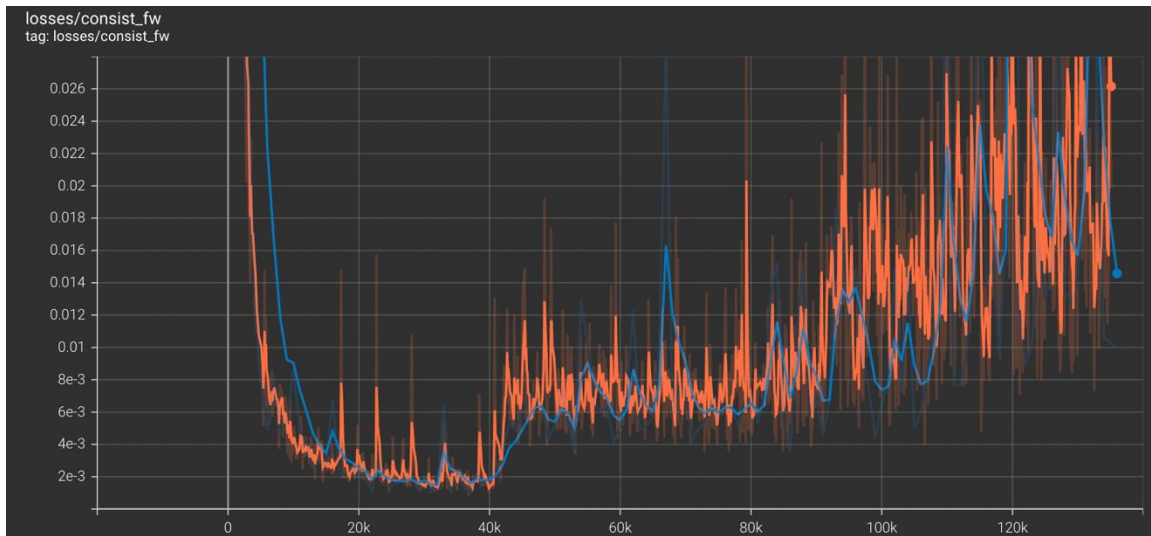
logs/train

tag: tower_0/proj_img_fw/image

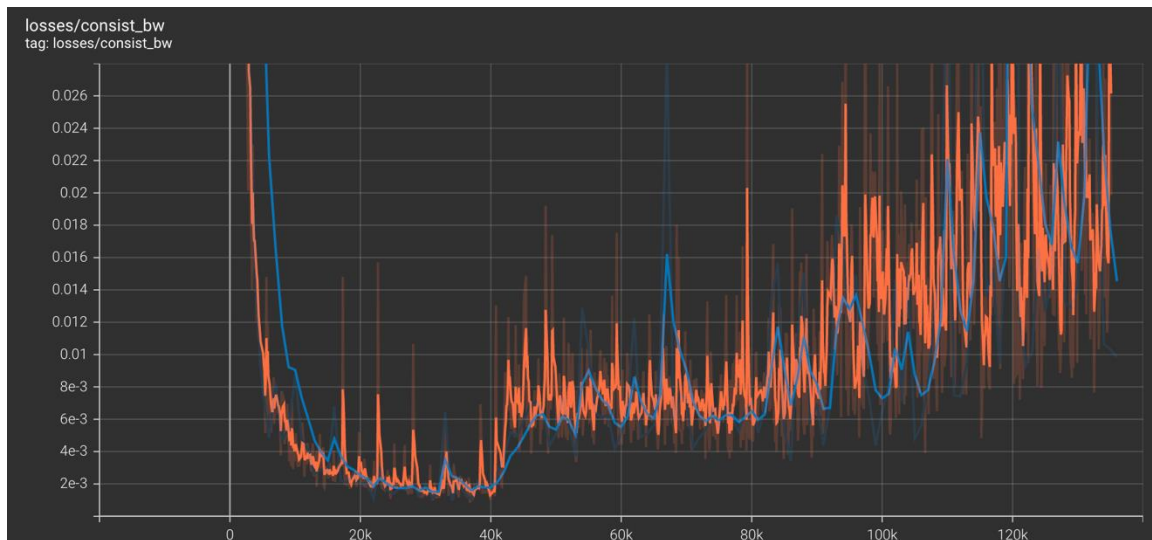
step 135,100 Fri Jan 21 2022 14:57:25 GMT+0200 (Eastern European Standard Time)



- *Forward flow-consistency loss*

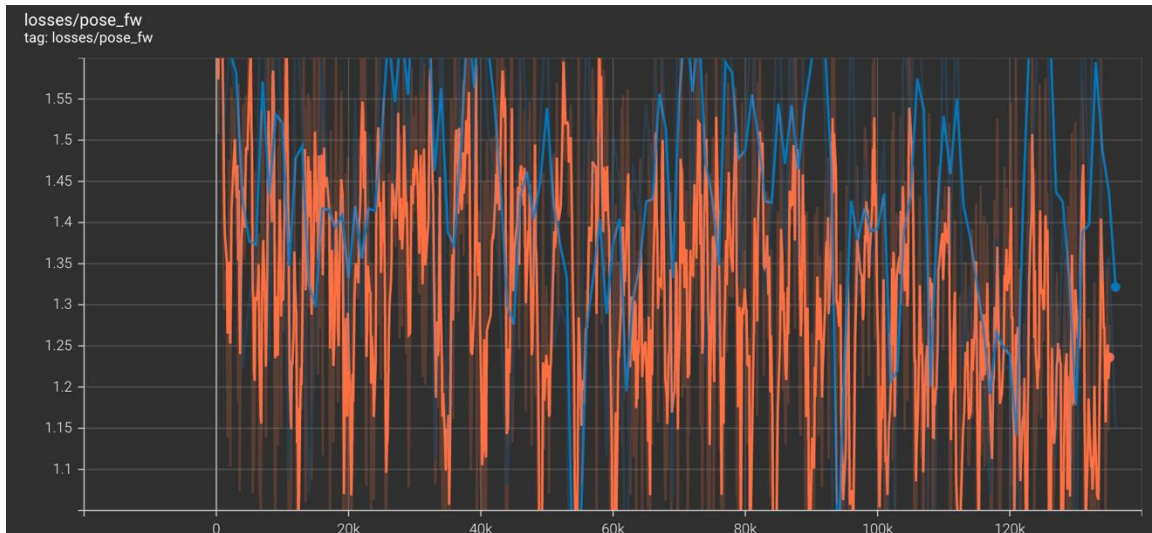


- *Backward flow-consistency loss*

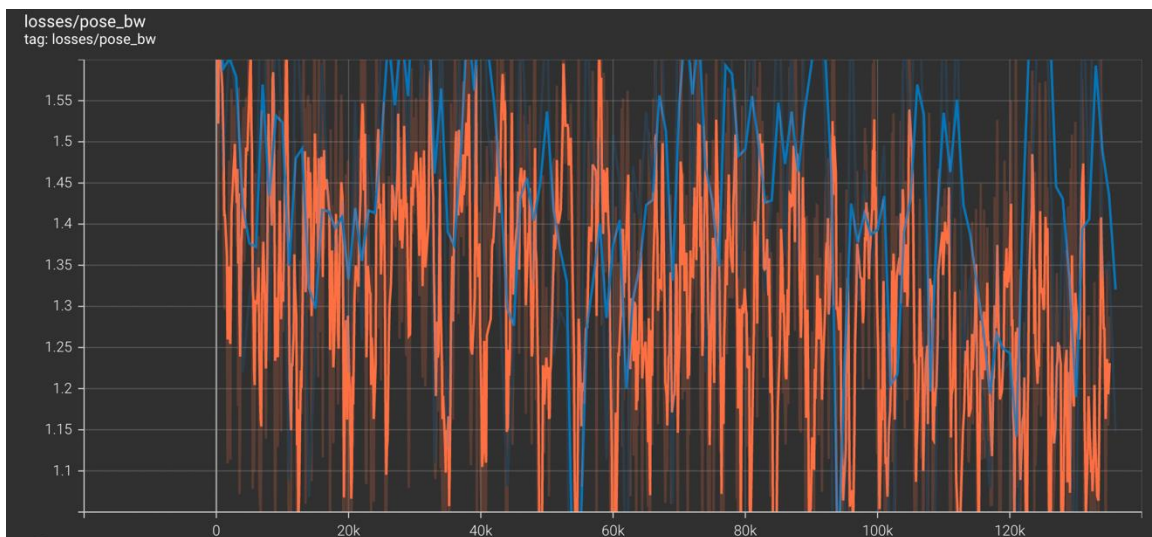


Pose Estimation

- *Forward pose estimation loss*



- *Backward pose estimation loss*




Occlusion Mask

- *Forward occlusion mask*

tower_0/fw_occ_mask/image

tag: tower_0/fw_occ_mask/image


step **136,000**Fri Jan 21 2022 07:44:47 GMT+0200 (Eastern European Standard Time)



tower_0/fw_occ_mask/image

tag: tower_0/fw_occ_mask/image

step **135,100**Fri Jan 21 2022 14:57:25 GMT+0200 (Eastern European Standard Time)



- *Backward occlusion mask*

tower_0/bw_occ_mask/image

logs/eval

tag: tower_0/bw_occ_mask/image

step **136,000** Fri Jan 21 2022 07:44:47 GMT+0200 (Eastern European Standard Time)



tower_0/bw_occ_mask/image

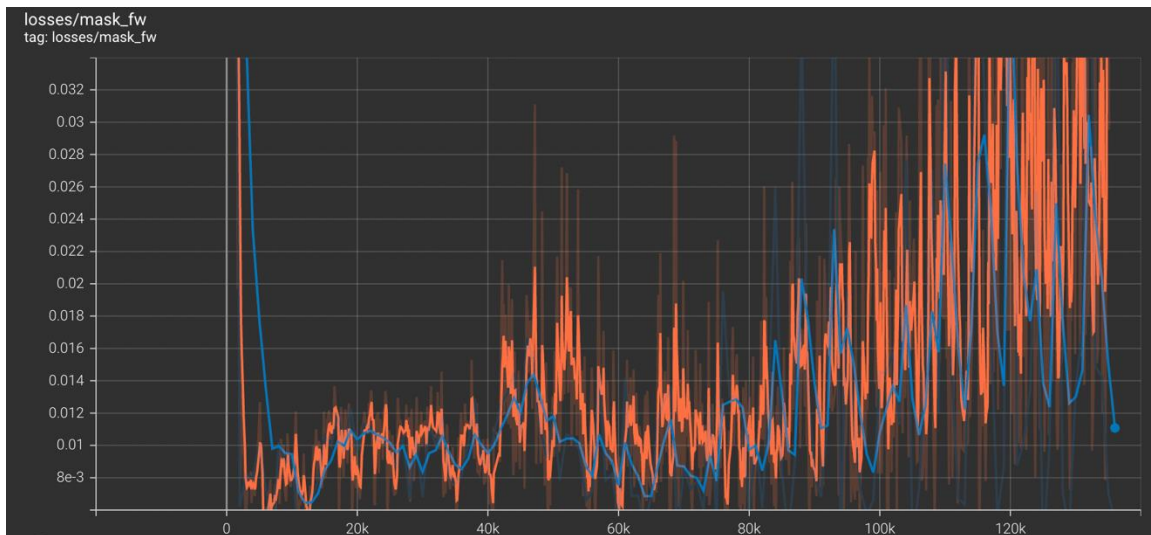
logs/train

tag: tower_0/bw_occ_mask/image

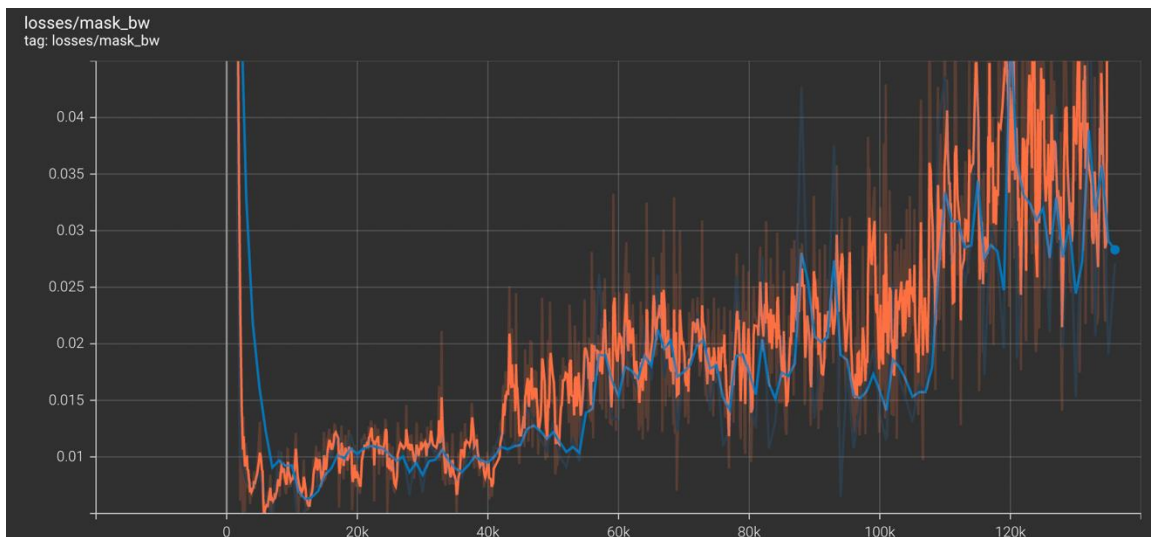
step **135,100** Fri Jan 21 2022 14:57:25 GMT+0200 (Eastern European Standard Time)



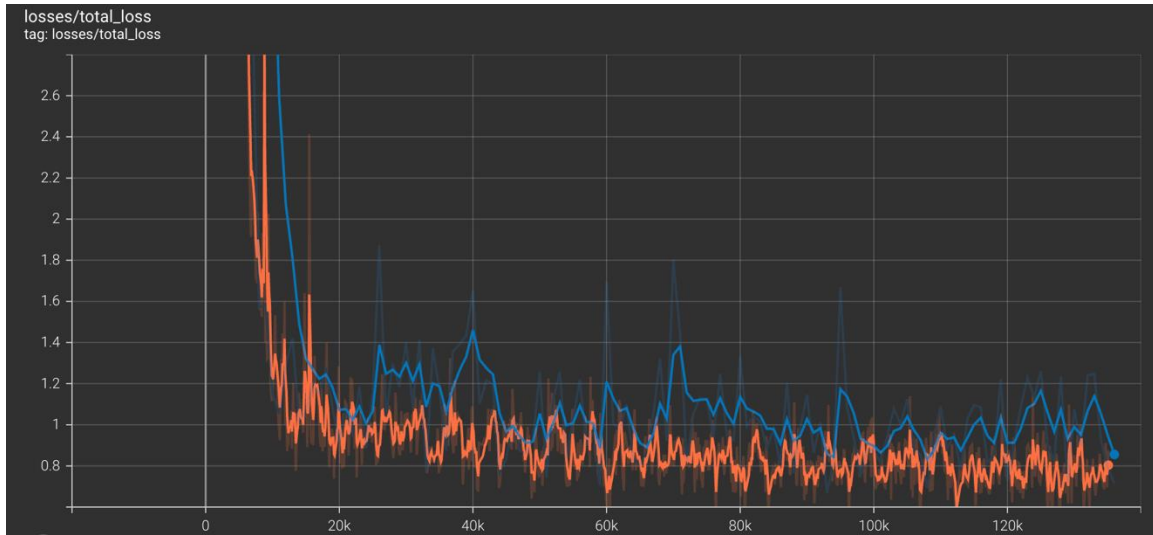
- *Forward mask loss*



- *Backward mask loss*



Total Loss



OBSERVATION

The model needs more training in order to reproduce the paper's results. We noticed that the depth estimation and the image reprojection from qualitative judgement are reaching good reprojection of the images and good depth estimation. The total loss is continuing to decrease.

WHAT IS NEXT?

We need to continue training for 10 to 20 epochs to reach the paper results as some losses need more training to be minimized than others as we found from monitoring the loss curves throughout the 27 epochs that we trained.

PROPOSED MODIFICATIONS

1. *Initializing the ConvLSTM hidden state instead of initially zero.*
2. *Removing losses.*
3. *Adding losses.*
4. *Architecture modification:*

*For example, we can try deeper depth and pose estimation network
Change the number of input sequence to account for the big
difference in frames in the flow consistency network.*

5. *Changing the training procedure:*

*We can freeze some parts of the network for some epochs and then
continue to train them after the first parts stabilize.*