Learning the Structure from Motion — An Unsupervised Approach

Khoi Viet Pham
PhD Student, Computer Science
Email: khoi@terpmail.umd.edu

Gnyana Teja Samudrala
M.Eng, Robotics
Email: sgteja@terpmail.umd.edu

John D. Kanu
PhD Student, Computer Science
Email: jdkanu@terpmail.umd.edu

Abstract—**In this paper we present an unsupervised learning framework for predicting monocular depth and ego-motion from video sequences. We try to improve the end-to-end learning approach presented in [1]. In contrast to the original SfMLearner our training data set is limited to a subset of the original KITTI data set. Our method uses ResNet architecture for depth estimation, with SSIM and forward-backward consistency loss which are adapted from [4]. The evaluations and comparision against provided KITTI data set show the effectiveness of our approach, which improves the *abs_rel* depth metric of SfMLearner by about 18% on the KITTI depth benchmark.**

I. Introduction

Developing a 3D scene geometry from a video sequence is the fundamental problem in perception. It involves various vision tasks like feature matching, depth estimation and ego-motion. These techniques have a wide range of applications in autonomous driving, interactive robotics, localization and navigation, etc. Traditional methods use a integrated pipeline of above tasks to simultaneously recover depth and pose of the camera. But these methods are prone to outliers in non-textured regions which cannot be completely avoided.

In order to overcome these problems, we apply deep learning models for each task and achieve remarkable developments over traditional methods. But in order to develop these models we need large data sets with labels to perform it in a supervised way. To avoid that we implement unsupervised framework, which is based on warping nearby views to the target using the computed values of depth and the pose.

II. Related work

Structure-from-Motion (SfM) is the problem of inferring scene structure and camera motion from images. After a review of recent literature on this topic, we have identified several methods that have inspired our work in this paper. Zhou et al. [1] jointly train two convolutional neural networks for predicting depth and pose, with view synthesis as supervision (*SfMLearner*). Using a similar approach, Yin et al. [4] train convolutional networks predicting depth and pose for rigid reconstruction, along with a residual flow network that predicts optical flow for non-rigid motion localization (*GeoNet*). An adaptive geometric consistency loss is employed, improving robustness to outliers. Zou et al. [5] propose an unsupervised learning framework for predicting single-view depth, pose, and optical flow, with forward-backward consistency loss and edge-aware smooth loss (*DF-Net*).

III. Approach

We present several modifications to SfMLearner that substantially improve accuracy. Modifications include the replacement of the DispNet with a residual network, enhancement of the learning signal through the integration of forward-backward warping consistency loss and structural similarity loss, and photometric augmentation for improved generalization. These modifications improve the *abs_rel* depth metric of SfMLearner by relatively 18% on the KITTI depth benchmark.

A. Network Architecture

Similar to the SfMLearner architecture, we employ two networks for predicting depth and pose. Rather than using a DispNet to predict depth, we employ ResNet50 [2], a 50-layer deep residual architecture that already achieves superb performance on image classification task. The ResNet architecture offers the following advantages: (1) skip-connection to allow easier gradient backpropagation and mitigate the vanishing gradient problem (2) deep architecture with residual paths make it a complex ensemble of shallower networks [3] that is highly powerful to this task.

The idea of using ResNet is also employed in other works in this literature, such as GeoNet [4] and DFNet [5]. However, their implementations are not exactly the same as the originally proposed ResNet from [2]. For example, in the first residual block, GeoNet & DFNet's implementations employ a stride of 2, whereas the original ResNet does not. In this work, we stick to the same architecture as in the originally proposed ResNet and we refer to it as DispResNet.

We also notice the SfMLearner network experiences overfitting on the provided KITTI training set of 12K images. Therefore, we simplify its DispNet architecture by reducing the number of feature maps in each layer. We refer to this network as DispNetSimple.

Since our primary objective is to improve the estimated depth map, we tended to focus our efforts on improving the depth prediction module, and left the Pose CNN intact.

B. Formulation of the Loss Function

1) Forward-backward warping consistency loss: The original view synthesis objective from SfMLearner already imposes a geometric forward warping constraint, i.e. the model warps each source image to the target coordinate frame using the predicted depth from the depth module, and attempts to minimize the L1 loss between the target frame and the warped frame (more details can be found in section 3.1 from [1]). Inspired from the work of GeoNet [4], in addition to this forward warping constraint, we add an extra backward warping constraint by warping the target frame back to each source coordinate frame and try to minimize the L1 loss between them. This idea is presented in the last paragraph of section 3.4 in [4].

We refer to this as the forward-backward warping consistency loss function, which imposes a geometric consistency constraint on the depth map estimated from the target frame as well as each source frame. By enforcing forward-backward warping consistency on the depth prediction, we identify invalid regions and impose a constraint on valid regions, encouraging the network to produce consistent predictions for the forward and backward directions. The forward and backward loss is formulated as the following objective

$$\mathcal{L}_{fw} = \rho(I_t, \tilde{I}_s^{s\to t}), \tag{1}$$

$$\mathcal{L}_{bw} = \rho(I_s, \tilde{I}_t^{t\to s}), \tag{2}$$

where $\rho(.)$ is a dissimilarity metric (e.g. L1 loss), I_s and I_t are the source and target frame, $\tilde{I}_s^{s\to t}$ is the result of warping the source frame to the target coordinate frame, and $\tilde{I}_t^{t\to ts}$ is vice-versa. Here, we want to train the model to minimize this dissimilarity between all pairs of source and target frame.

2) Image dissimilarity metric: We employ the L1 loss as our main image dissimilarity metric. Apart from it, we also incorporate a structural similarity index (SSIM) into the dissimilarity metric $\rho(.)$, given by [7]. The SSIM is the comparison between two images on various windows of equal size. The index is a value between -1 and 1, where 1 is the case in which both images are identical. It is defined as follows,

$$SSIM = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2) + c_2}, \tag{3}$$

where μ is the average, σ is the covariance, $c1$ and $c2$ are the variables to stabilize the division with weak denominator. In overall, our dissimilarity function $\rho(.)$ is

$$\rho(I_a, I_b) = \lambda_{l1}\|I_a - I_b\|_1 + \lambda_{ssim}SSIM(I_a, I_b). \tag{4}$$

3) Smooth function: : We use the same smooth function as in SfMLearner.

Our final objective function is

$$\mathcal{L}_{final} = \sum_l \sum_{\langle t,s\rangle} \mathcal{L}_{fw} + \mathcal{L}_{bw} + \lambda_s \mathcal{L}_{smooth}, \tag{5}$$

where l iterates over 4 image scales, $\langle t, s\rangle$ iterates over all source-target pairs.

C. Data Augmentation and pre-processing

In addition to random scaling and cropping as in SfM-Learner, images are pre-processed and augmented using the method presented with UnFlow [6]. Images are pre-processed with random additive Gaussian noise ($0 < \sigma \leq 0.04$), random additive brightness changes, random multiplicative color changes ($0.9 \leq m \leq 1.1$), contrast (from [-0.3, 0.3]) and gamma changes ([0.7,1.5]). All inputs are normalized. The original SfMLearner modified with data augmentation is addressed in the table as SfMLearner_aug.

Set	Model	Error metrics			
		Abs Rel	Sq Rel	RMSE	RMSE log
Train	SfMLearner (ours)	0.253	6.512	7.684	0.295
Train	SfMLearner_aug	0.215	1.992	5.657	0.294
Train	DispNetSimple	0.185	3.714	6.465	0.249
Train	DispResNet	0.207	4.813	6.642	0.266
Test	SfMLearner (ours)	0.207	2.007	7.043	0.288
Test	SfMLearner_aug	0.225	2.136	5.822	0.303
Test	DispNetSimple	0.201	1.839	6.776	0.279
Test	**DispResNet**	**0.169**	**1.324**	**6.295**	**0.252**

TABLE I: Single-view depth error metrics results on the provided 12K KITTI training set and the KITTI testing set acquired using the split of Eigen et al [8]. Our best model is highlighted.

IV. Experiments

A. Training Details

We train on the 12K provided images from the KITTI dataset. Each image is 128×416. Explainability regularization weight was left at 0. Networks were optimized using the Adam optimizer with learning rate 2×10^{-4}, $\beta_1 = 0.9, \beta_2 = 0.999$ and batch size 4, which were identified by a parameter sweep on the training data.

Regarding hyperparameters in the loss function, emperically, we we select $\lambda_{l1} = 0.15, \lambda_{ssim} = 0.85, \lambda_s = 0.5$.

B. Network Architecture

Regarding DispNet and PoseNet, we use the same architecture as in SfMLearner. For DispResNet, we use ResNet50 (exact same architecture as in [2]) for the encoder, where as the decoder is similar to that of DispNet (with some small modifications which can be seen from the source code). We use ReLU as the activation function in all networks.

V. Results

A. Single-view depth estimation

We retrain the SfMLearner for 200K iterations on the provided 12K images and report the experimental results in table I and II. Note that we could not reproduce the results reported by the authors on their GitHub, even though we have used the same source code except for some modifications to the *collect_summaries* function. However, our results are similar to their version in the SfMLearner paper [1]. We also show the results achieved by DispResNet and DispNetSimple in table I and table II. Note that these models are trained on the provided 12K training images, with data augmentation as mentioned in the previous section, and tested on the KITTI testing set split of Eigen et al [8].

From table I, we notice some strange results: for *SfM-Learner (ours)* and *DispResNet*, their training error metrics are both worse than their test error metrics (abs rel, sq rel, rmse,

Set	Model	Accuracy metrics		
		$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Train	SfMLearner (ours)	0.748	0.882	0.933
Train	SfMLearner_aug	0.700	0.893	0.954
Train	DispNetSimple	0.804	0.930	0.968
Train	DispResNet	0.789	0.925	0.964
Test	SfMLearner (ours)	0.696	0.889	0.953
Test	SfMLearner_aug	0.679	0.886	0.950
Test	DispNetSimple	0.702	0.895	0.957
Test	**DispResNet**	**0.751**	**0.913**	**0.967**

TABLE II: Single-view depth accuracy metrics results on the provided 12K KITTI training set and the KITTI testing set acquired using the split of Eigen et al [8]. Our best model is highlighted.

Model	Seq. 09	Seq. 10
SfMLearner (ours)	$\mathbf{0.0178 \pm 0.0068}$	$\mathbf{0.0141 \pm 0.0091}$
DispNetSimple (ours)	0.0183 ± 0.0068	0.0143 ± 0.0096
DispResNet (ours)	0.0185 ± 0.0064	0.0145 ± 0.0089

TABLE III: Absolute Trajectory Error (ATE) on the KITTI odometry split, sequence 9 and 10, averaged over all 3-frame snippets.

rmse log). Normally, we expect the model performance on the training set to be much better than the performance on the testing set, however, this doesn't happen here. We did not have time to inspect this problem in more details, however, we still have some thoughts to explain this phenomena. Table II shows that the accuracy metrics on the training set is still clearly better than that of the testing set; therefore, we suspect that there are many outliers in the training set that greatly affect the error metrics while they only have minor effect on the accuracy metrics (the accuracy metric is based on the number of estimated depth points that fall into 3 specific ranges, so it is less prone to outliers).

From table I and II, we see that the *DispResNet* outperforms the other models (improve SfMLearner's abs_rel metric by $\approx 18\%$). Figure 1 shows the abs_rel evaluation results versus training iterations for the 3 models: SfMLearner, DispNetSimple, and DispResNet. The plot shows that the performance of DispResNet is still improving, however, we did not have enough time to pursue more training.

Figure 3 shows a heatmap visualization of approximate depth prediciton error for difficult cases. We can observe much of the error comes from moving objects, thin structures, and complex objects like trees and bushes. One interesting case is included, showing curved lines that resemble optical flow for a camera rotating around the vertical axis, which is an interesting phenomenon to inspect further.

B. Pose estimation

We evaluate the pose estimation over 3-frame snippets (not 5-frame snippets as in [1]). From III, we see that *SfMLearner* has a slightly better result than our *DispResNet*. Since we did not focus on optimizing for pose, we could not give any explanation for this part.

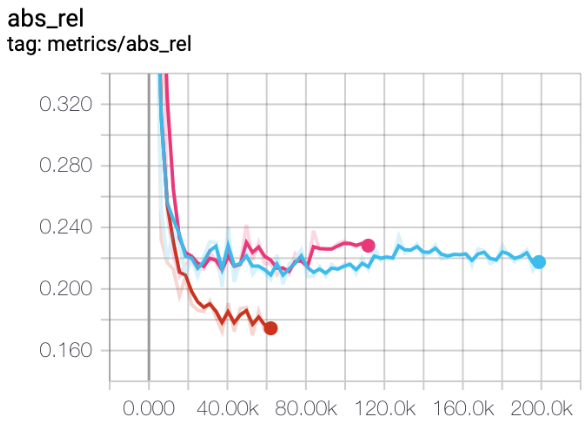


Fig. 1: Plot of abs_rel versus training iterations for our 3 models: SfMLearner (pink), DispNetSimple (blue), and DispResNet (red).

VI. Discussion

We would like to discuss more the reason we select ResNet as our depth module. After exploring other related works, we believe all variants of the forward-backward consistency loss offer great performance improvement over SfMLearner. Therefore, our first attempt was to train SfMLearner with the proposed forward-backward consistency loss. However, from inspecting the training/validation plot, we believe the SfMLearner is not powerful enough to learn to impose the forward-backward consistency constraint. Therefore, we approach to select the ResNet architecture as our depth module due to its well-known excellent performance on other computer vision tasks. Experimental results have shown that our intuition is indeed correct, the *DispResNet* performance clearly outperforms SfMLearner on the depth metric.

References

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.
[2] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)
[3] A. Veit, M. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow network. In NIPS, 2016.
[4] Z. Yin and J. Shi. GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[5] 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, 2018.
[6] S. Meister, J. Hur, and S. Roth. "UnFlow: Unsupervised learning of optical flow with a bidirectional census loss," in Thirty-Second AAAI Conference on Artificial Intelligence. 2018.
[7] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. TIP, 2004.
[8] 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.

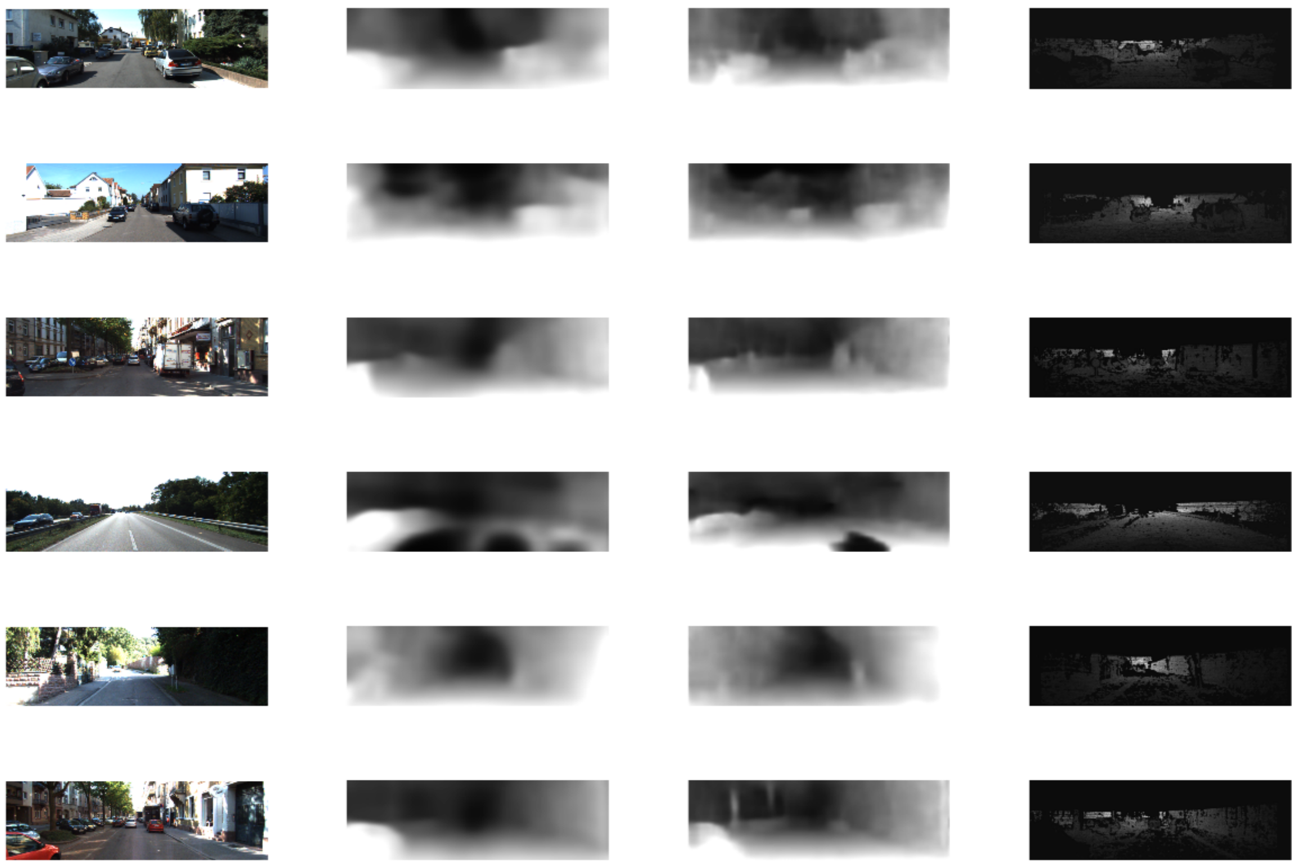
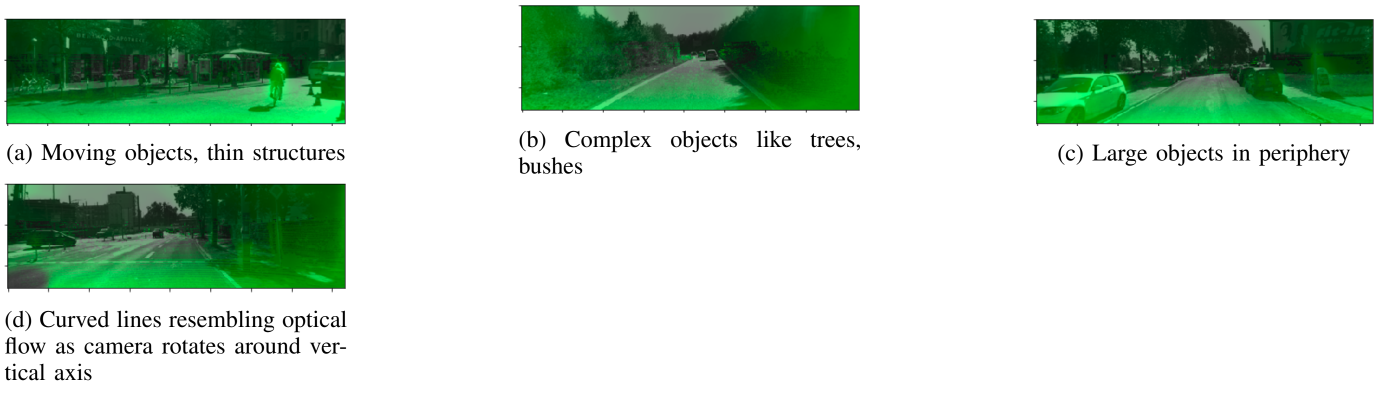


Fig. 2: The figure showing depth maps of SfMLearner in 2 column and DispResNet in 3 column. The 4 column is the ground truth map. The first 3 rows are the data in training set and last 3 correspond to validation set.



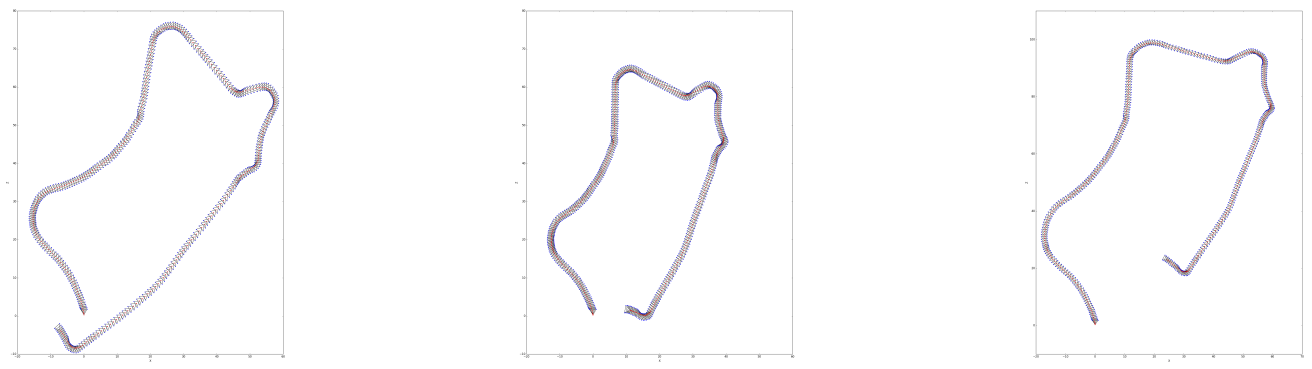
(a) Moving objects, thin structures

(b) Complex objects like trees, bushes

(c) Large objects in periphery

(d) Curved lines resembling optical flow as camera rotates around vertical axis

Fig. 3: Visualization of difficult cases using approximate depth error heat maps

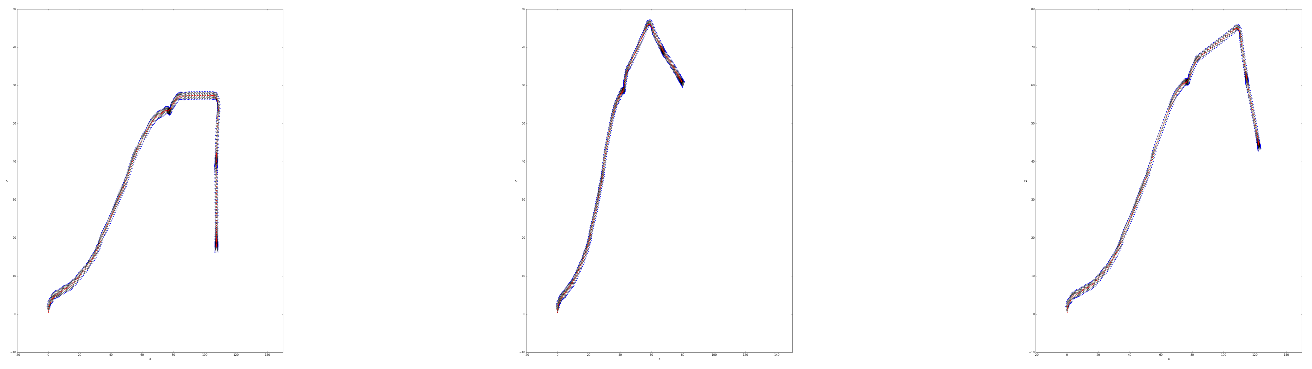


(a) Ego motion estimated from SfMLearner

(b) Ego motion estimated from DispNetSimple with SSIM loss

(c) Ego motion estimated from DispResNet

Fig. 4: Ego motion for sequence 9. The blue points are the face of the camera and the red points are centre point.



(a) Ego motion estimated from SfMLearner

(b) Ego motion estimated from DispNetSimple with SSIM loss

(c) Ego motion estimated from DispResNet

Fig. 5: Ego motion for sequence 10. The blue points are the face of the camera and the red points are centre point.