Project 3 - Structure from Motion

Nikhil Mehra and Sayan Brahma
M.Eng Robotics
University of Maryland - College Park
nmehra@terpmail.umd.edu | sbrahma@terpmail.umd.edu

***Abstract*—In this project we present a traditional as well as a deep learning approach for structure from motion (SfM). Reconstructing a 3D scene and simultaneously obtain the camera poses of a monocular camera w.r.t. the given scene is known as Structure from Motion (SfM). Part two of the project consists of improving an existing 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 [2]. In contrast to the original SfMLearner our training data set is limited to a new dataset given instead of the original KITTI data set. Our method does not change the architecture much for depth estimation instead alters the way of input, with SSIM and forward-backward consistency loss which are adapted from [1]. The evaluations, testing and comparison against the KITTI data set show the effectiveness of our approach, which improves the abs rel depth metric of SfMLearner.**

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 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. Traditional Approach

A. Estimate Fundamental Matrix

The Fundamental matrix, F is only an algebraic representation of epipolar geometry. In laymen terms it is the relationship between the correspondences in the given stereo images. It is a 3x3 matrix with 9 components that can be defined by the following equation, where x_1 and x_2 are the coordinates of a feature correspondences in the given two images.

$$x_1^T F x_2 = 0$$

$$\begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx_m' & x_my_m' & x_m & y_mx_m' & y_my_m' & y_m & x_m' & y_m' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

$$Ax = 0$$

The above equation requires $N \geqslant 8$ point correspondences and can be answered by solving the linear least squares using Singular Value Decomposition (SVD). Then the fundamental matrix is enforced with the rank 2 constraint, the last singular value of the estimated F is set to zero. As the features computed through SIFT can be noisy RANSAC is applied to reject the outliers. Hence we get refined correspondences (inliers) as shown in Figures 1 and 2. The Fundamental matrix was thus then calculated with the eight points which contains the maximum number of inliers. The points were normalized and the 8-point algorithm was used to solve for the matrix. The computed fundamental matrix was then denormalized to get the correct matrix.

B. Estimate Essential Matrix

Essential Matrix, E is also a 3×3 matrix, but with some additional properties that relates the corresponding points assuming that the cameras obeys the pinhole model (unlike F).

$$E = K^T F K$$

where K is the calibration matrix. The Essential matrix is corrected by reconstructing it with (1,1,0) singular values.

$$E = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$$

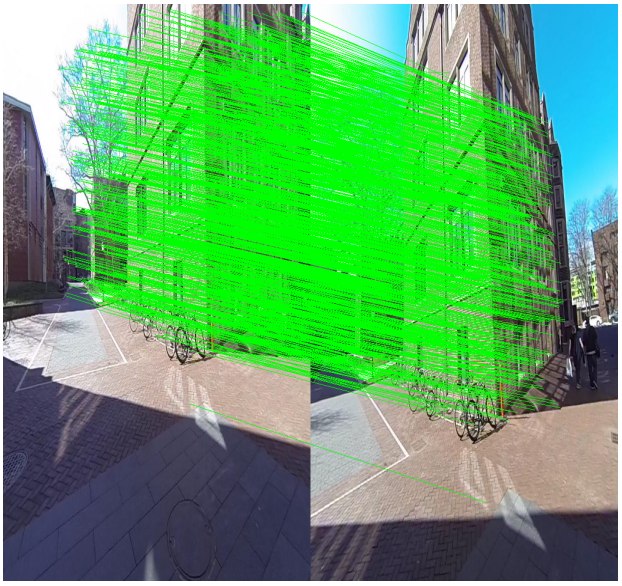


Fig. 1: SIFT Features

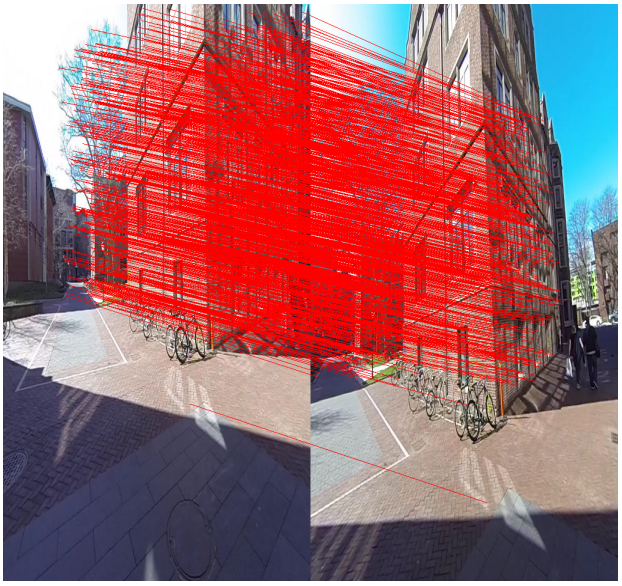


Fig. 2: Features after RANSAC

C. Estimate Camera Pose from Essential Matrix

We assume that the first camera is located at the origin and is aligned with of the world coordinate system. We get four different camera poses by decomposing essential matrix using singular value decomposition.

$$E = UDV^T$$

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1) $C_1 = U(:,3)$ and $R_1 = UWV^T$
2) $C_2 = -U(:,3)$ and $R_2 = UWV^T$
3) $C_3 = U(:,3)$ and $R_3 = UW^TV^T$
4) $C_4 = -U(:,3)$ and $R_4 = UW^TV^T$

It was made sure that the determinants of rotation matrices were positive. If not, the camera pose was corrected by multiplying it with -1.

D. Triangulation Check for Cheirality Condition

All the feature correspondences are first linearly triangulated through each of the four different camera poses we received from the last step. The points are triangulated between the camera first and second by solving the below equation using SVD.

$$\begin{bmatrix} \begin{bmatrix} x_1 \\ 1 \end{bmatrix}_x P_1 \\ \begin{bmatrix} x_2 \\ 1 \end{bmatrix}_x P_2 \end{bmatrix} X = 0$$

where P_1 and P_2 are the projection matrix of the firt and second camera respectively.

$$P = KR\left[I|-C\right]$$

To obtain disambiguate the right camera pose from the four different camera poses that we computed in the last step. We used cheirality condition, which states that any point projected in the world frame using some camera poses must lie in the front of the camera. i.e. $r_3\left(X - C\right) > 0$. where r_3 is the z-axis of the camera rotation matrix. So the combination of (R,C) that produces maximum number of linearly triangulated points in front of the camera is chosen to be the right values of R and C.

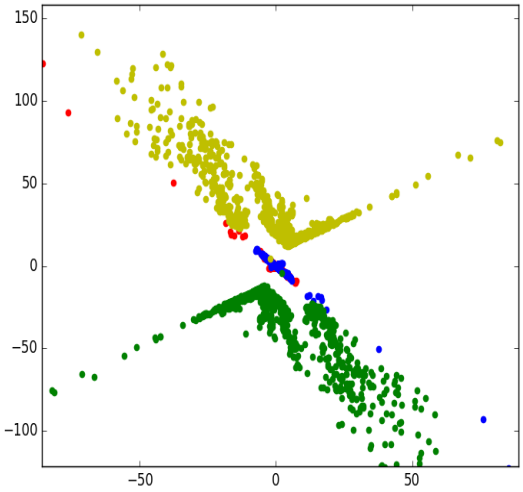


Fig. 3: Linearly triangulated points plotted for different camera poses

Through the linear triangulation we minimized the algebraic error in the triangulation, to minimize the geometric error we use the non-linear triangulation by using the obtained camera pose and the linearly triangulated 3D projections. Non-linear

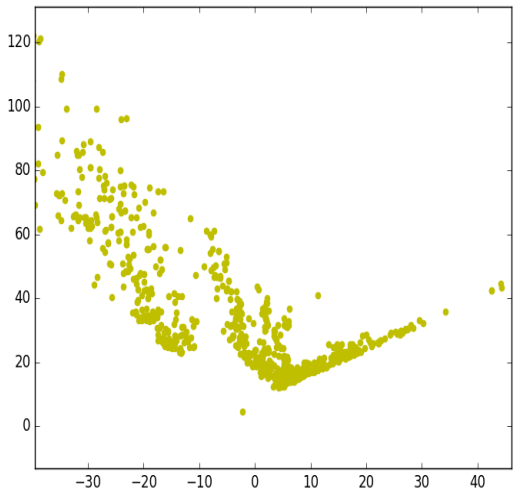


Fig. 4: Linearly triangulated points plotted after disambiguation of camera pose (true camera pose)

triangulation is nothing but a non-linear optimization of the reprojection error defined as:

$$\min_x \sum_{j=1,2} \left(u^j - \frac{P_1^{jT}\tilde{X}}{P_3^{jT}\tilde{X}} \right)^2 + \left(v^j - \frac{P_2^{jT}\tilde{X}}{P_3^{jT}\tilde{X}} \right)^2$$

The optimization of the above function is done by using scipy.optimize.least_squares

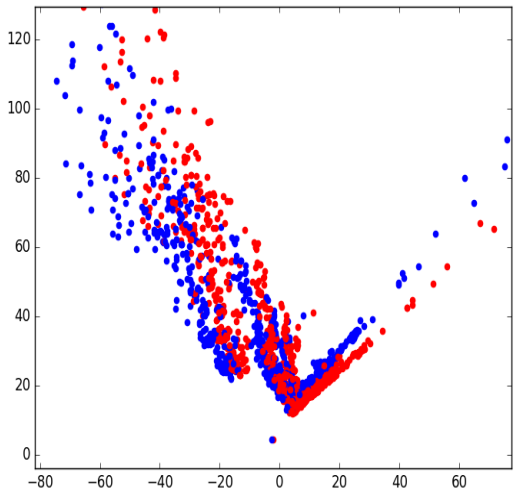


Fig. 5: Non-Linearly triangulated points(blue) versus linealy traingulated points(red)

E. Perspective-n-Points

Since we have a set of n 3D points in the world, their 2D projections in the image and the intrinsic parameter; the 6 DOF camera pose can be estimated using linear least squares. This fundamental problem, in general is known as Persepective-n-Point (PnP).

1) Linear PnP: First, we need to find a rotation and translation matrices for camera 3-6 that would satisfy a set of real world points projected to a set of image points from images 3-6 respectively. First we normalized the image points by multiplying with the inverse of the camera calibration matrix.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K^{-1}x$$

$$\begin{bmatrix} 0_{1\times4} & -\tilde{X}^T & v\tilde{X}^T \\ \tilde{X}^T & 0_{1\times4} & -u\tilde{X}^T \\ -v\tilde{X}^T & u\tilde{X}^T & 0_{1\times4} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = 0$$

We need $N \geqslant 6$ pair of image points and the corresponding world points. The above equation can be solved using SVD. We get the rotation and translation vector from the last row of the V^T matrix.

2) PnP RANSAC: PnP is prone to error as there are outliers in the given set of point correspondences. Hence we use RANSAC. We get inliers by checking the reprojection error below a certain threshold. In our case we have set the threshold to be of 10.

$$e = \left(u - \frac{P_1^T\tilde{X}}{P_3^T\tilde{X}} \right)^2 + \left(v - \frac{P_2^T\tilde{X}}{P_3^T\tilde{X}} \right)^2$$

3) Non-Linear PnP: In this we minimize the reprojection error by optimizing the rotation, translation given the 3D to 2D correspondences found using non-linear triangulation. This task is also a non-linear optimization and we have used scipy.optimize.least_squares function for it. A key point to keep in mind is that the parameters to be optimized are only the rotation matrix and translation vector. If we give projection as an input to the optimizer it will try to optimize the calibration matrix embedded into the projection matrix too, which ultimately leads to poor results. Also, a compact representation of the rotation matrix like quaternion is a better choice to enforce orthogonality of the rotation matrix while optimization. Hence the rotation matrix was converted into a quaternion format beforehand. Hence we give 7 input values to the optimizer, 4 of quaternion and 3 of the translation vector to the optimizer. We convert the quaternions back to the rotation matrix after the optimization.

F. Visibility Matrix

We created a 2D matrix of size (No.Points x 6) and a list which stores the different world points. Index of any world point is similar to the row index of that point in the 2D matrix. A dictionary is also prepared which stores mapping from the 2-D image coordinates to the 3-D world coordinates indexes.

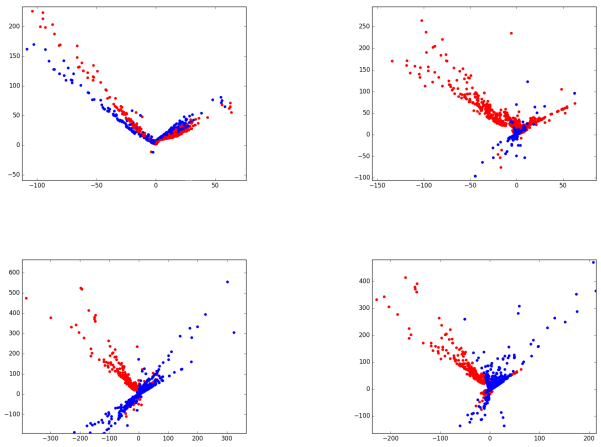


Fig. 6: Plots are finding camera poses using Non-Linear PnP

G. Bundle Adjustment

Bundle adjustment is defined as the problem of simultaneously refining the 3D coordinates describing the scene geometry, the parameters of the relative motion, and the optical characteristics of the camera/cameras employed to acquire the images, according to an optimality criterion involving the corresponding image projections of all points. In laymen terms it is also a no-linear optimization of the whole process. As the variables to be optimized are too many, we need to find a way to optimize the more important variables and ignore the less important ones. This can be done by constructing a sparse jacobian matrix which tells the optimizer function about the dependency of a residual on a particular variable, so that it does not spend time computing gradients for unrelated residuals and variables. This sparse jacobian matrix was sent as additional input to the scipy.least squares optimizer.

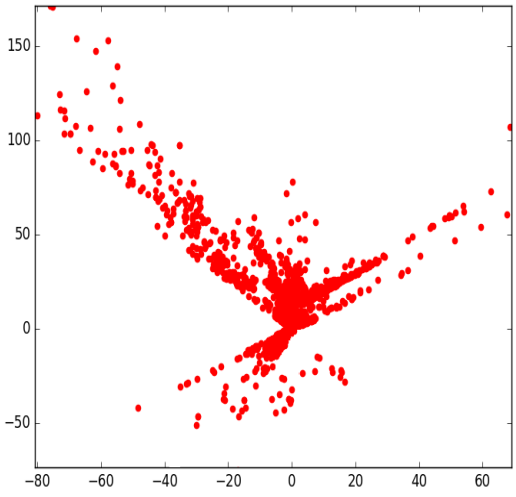


Fig. 7: Final Reconstruction after Bundle Adjustment

H. Discussion

Some key points to note are that the pipelines works best as long as RANSAC spits out the correct inliers to work with. We had to find a sweet spot for the threshold. Also as RANSAC is a random process, each iteration gave us a different result. Increasing and decreasing of the threshold both results in a poor performance, hence poor reconstruction down the line. Many feature points were repeated and hence were removed from the data set given to us. For future a single pandas file can be used for storing SIFT features. It will smooth out the PnP process. Code for Bundle Adjustment can be improved further as it didn't work good enough.

I. Results

Reconstruction for a large object like a building came out to be pretty good with some noise. Reconstruction of the smaller objects such a tree and bike came out to be really bad as compared to the building. But I believe that the deep learning approach for the structure of motion can give better results.

III. Deep Learning Approach

A. SfMLearner - Original

The SfMLearner Network 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.

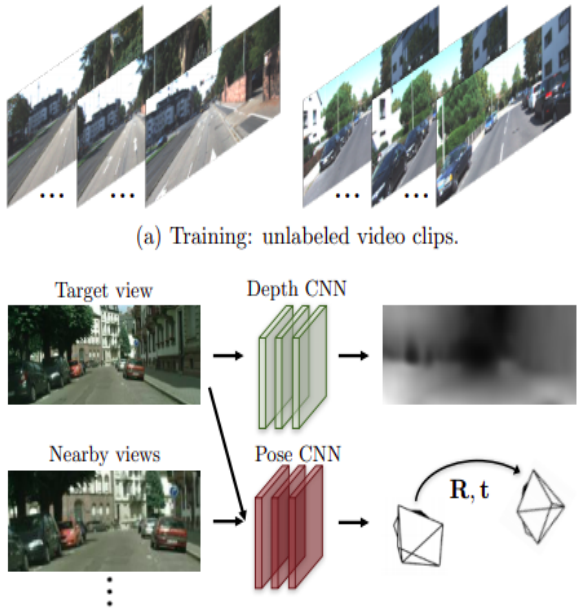


Fig. 8: The training data to this system consists solely of unlabeled image sequences capturing scene appearance from different viewpoints, where the poses of the images are not provided. The training procedure produces two models that operate independently, one for single-view depth prediction, and one for multi-view camera pose estimation.

The network consists of two different networks, each dedicated to training Depth and Pose respectively. The Depth network is realized using DispNet 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.

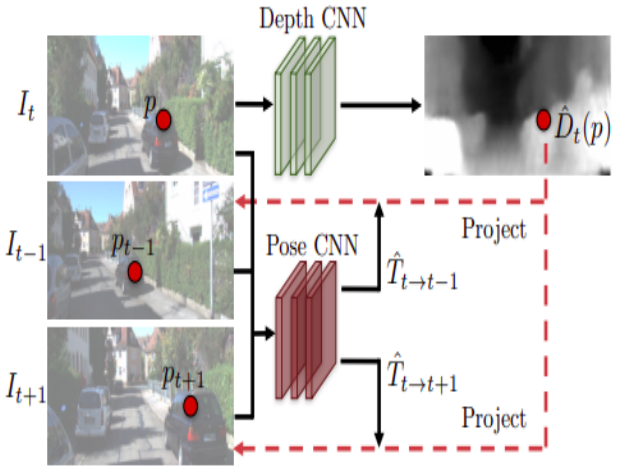


Fig. 9: Overview of the supervision pipeline based on view synthesis. The depth network takes only the target view as input, and outputs a per-pixel depth map $\hat{D}$. The pose network takes both the target view (It) and the nearby/source views (e.g., It1 and It+1) as input, and outputs the relative camera poses $\hat{T}_{t\to t1}, \hat{T}_{t\to t+1}$. The outputs of both networks are then used to inverse warp the source views to reconstruct the target view, and the photometric reconstruction loss is used for training the CNNs. By utilizing view synthesis as supervision, they were able to train the entire framework in an unsupervised manner from videos.

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 Es for each target source pair. Based on this predicted Es the view synthesis objective is weighted correspondingly. This is a nutshell, that summarizes the structure of the SfMLearner network. The network tends to converge after about 15K iterations on the dataset which was provided to us. For this project, we restrict ourselves with training and testing on dataset given to us.

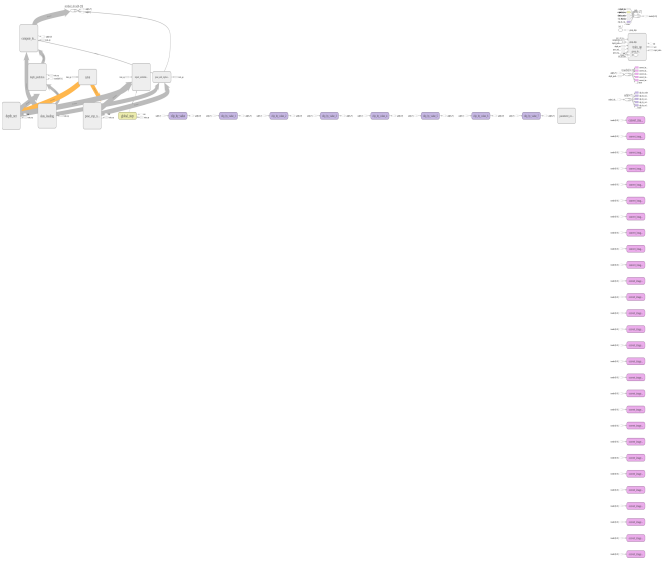


Fig. 10: Graphical summary of the network architecture

Fig. 11: Smoothed losses of the original Network

B. SfMLearner - Modified

1) Network architecture: Not much has been changed in the architecture. Similar to the SfMLearner architecture, we employ two networks for predicting depth and pose. 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 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 leverage 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.

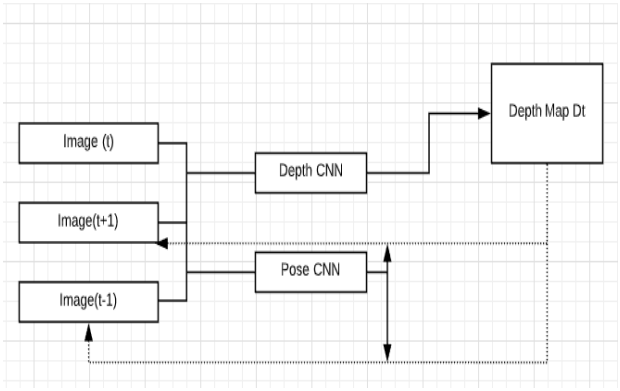


Fig. 12: Modified Network

2) Loss:

- 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. Inspired from the work of GeoNet [1], 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 [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.

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

  where $\rho(.)$ is a dissimilarity metric (e.g. L1 loss), Is and It 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 $\tilde{I}_t^{t \to s}$ is vice-versa. Here, we want to train the model to minimize this dissimilarity between all pairs of source and target frame.
- 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 reflection's, 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.

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

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

  where μ is the average, σ is the covariance, c1 and c2 are the variables to stabilize the division with weak denominator.

Fig. 13: SSIM Loss

- Smooth/Total loss: We use the same smooth function as in SfMLearner. Our final objective function is:

$$\mathcal{L}_{final} = \sum_{l} \sum_{<t,s>} \mathcal{L}_{fw} + \mathcal{L}_{bw} + L_{ssim}$$

where l iterates over 4 image scales, $< t, s >$ iterates over all source-target pairs.

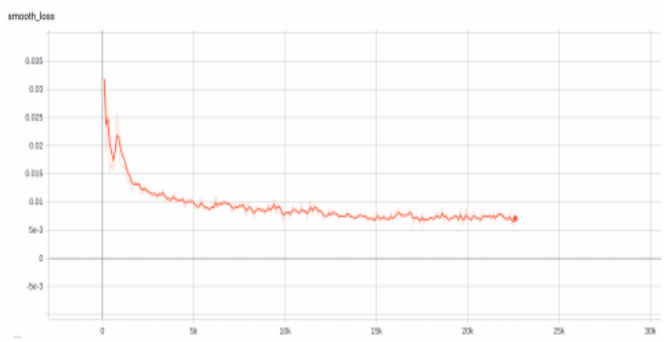


Fig. 14: Total Loss

3) 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. SfMLearner 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 and some additive Gaussian noise. These lead to improvements in loss.

4) 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 RTX 2060 GPU. The average time per iteration came out to be about 50 milliseconds per iteration, i.e., approx 6-7 hours for total training. The SfM learner took about 22 milliseconds but with a mini batch size of 4. Here is the Depth map comparison:

(a) Reference Image

(b) Depth prediction of SfMLearner

(c) Depth prediction of modified method

5) Experiments and Improvement scope:

- Batch Normalisation (negative impact)- With the goal of further regularizing the network, we also implemented batch normalization after every 4 convolutional layers.

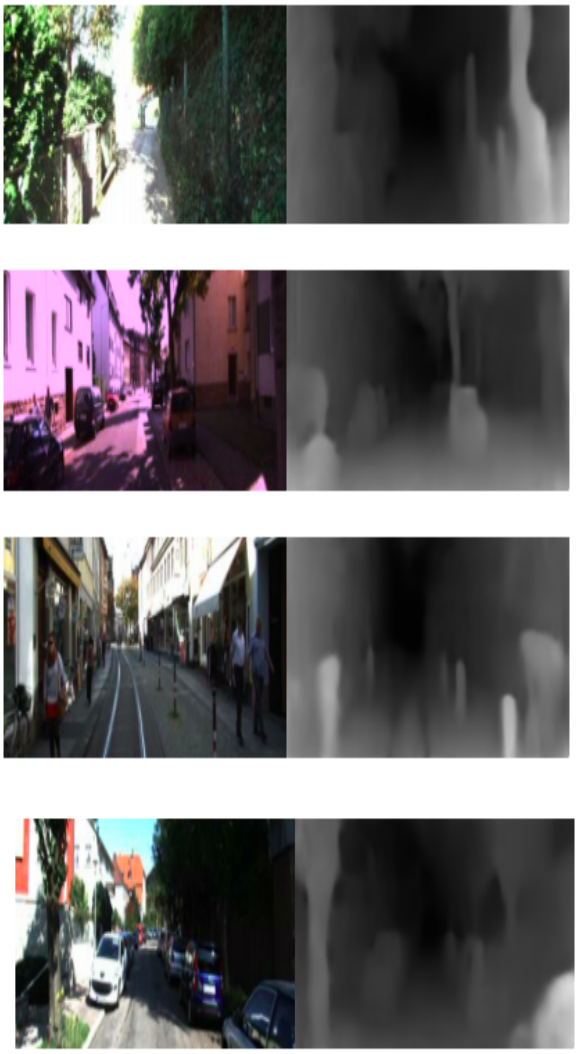
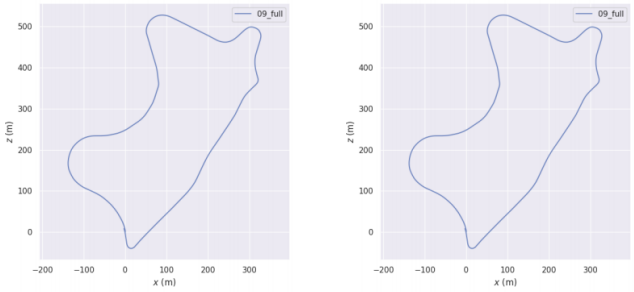


Fig. 15: Depth maps on the train dataset



(a) Ground truth trajectory for seq. 09

(b) Ground truth trajectory for seq. 10

Fig. 16: 2 Figures side by side

Unfortunately, it had a negative impact on the model and increased the overall loss. Hence, we decided to discard batch normalization from our implementation.

- RNN implementation (improvement scope) - Rather than using a DispNet to predict depth, we can employ RNN - LSTM. 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.

REFERENCES

[1] 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.

[2] Unsupervised Learning of Depth and Ego-Motion from Video By Tinghui Zhou*(UC Berkeley), Matthew Brown (Google), Noah Snavely(Google), David G. Lowe(Google)

[3] Microsoft Research Report - A Flexible New Technique for Camera Calibration By *Zhengyou Zhang - December 2, 1998*