CS231A : Final Report

Structure From Motion from 2 views

Naoshad Eduljee

Abstract : In this work I present a unsupervised learning framework for the task of monocular depth and camera pose estimation from unlabeled video sequences. The task of monocular depth and pose estimation are carried out using end to end learning method with new view synthesis as the supervisory signal. This method is completely unsupervised requiring only monocular video sequences for training. This approach uses a novel single view depth network with a loss function based on warping nearby views to the target using inferred depth and computed camera pose. The novel single view depth network is supervised by the loss function at training time but can be used separately for inference.
The novel single view depth network is based on a lightweight encoder-decoder network architecture to reduce computational complexity and latency.

Introduction: Inferring the 3D structure of a scene in a expedient manner so as to avoid obstacles while navigating an environment is a challenging unconstrained problem. We humans can easily perform such tasks without any effort. Classical methods such as ***Structure from Motion*** exist that require matching features/edges between the two images, then inferring motion and geometry from those matches. When images of the environment are taken in conditions of constant lighting, that have diffuse and rigid surfaces, and non-repeating visual texture, the matching process in well posed. But these techniques cause problems in areas of low texture, complex geometry/photometry, thin structures and occlusions. These methods are also very compute intensive and not suitable for realtime use. To address these issues one intuition we can take from our own ability to perform such tasks is of prior learning i.e. developing a structural understanding of the world through our past visual experiences gained by moving around and observing many scenes and developing mental models through observations. Recently deep learning methods have made significant progress in addressing these issues. Such methods require large amounts of data to train models but once trained they can be deployed in the field to infer scene geometry and camera motion in an expedient manner. Also recently methods have been developed that do not require the use of large corpus of labeled data. Using photometric reprojection loss between pairs on images of the same scene, one can train these neural networks in an unsupervised manner. I plan to use such an end-to-end deep learning method. Applications of this include autonomous driving, augmented/virtual reality, and robotics.

In our approach multiple views of a scene (multiple unlabeled images) are used as an implicit source of training data for single-view depth prediction, by utilizing view synthesis as a supervisory signal. Here the aim is to distill geometric reasoning capability from CNNs trained to perform warping-based view synthesis. The view synthesis pipeline acts as the inference procedure of a convolutional neural network, so that by training the network on large sequence of images taken consecutively or large-scale video data for the 'meta'-task of view synthesis, the network is forced to learn about intermediate tasks of depth estimation in order to come up with a consistent explanation of the visual world.

The overall approach is to use a deep learning framework for training ***a single-view depth CNN*** from unlabeled video sequences to infer the depth (parameterized as per-pixel depth maps under a reference view) and pose estimate (parameterized as 6-DoF transformation matrices) .

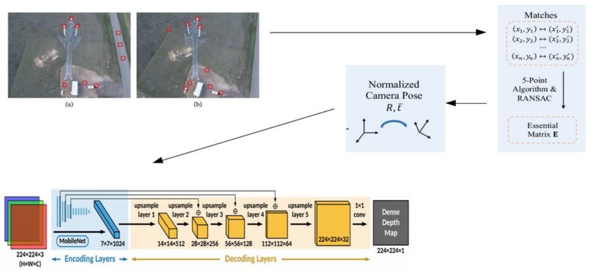
2. Related Work:

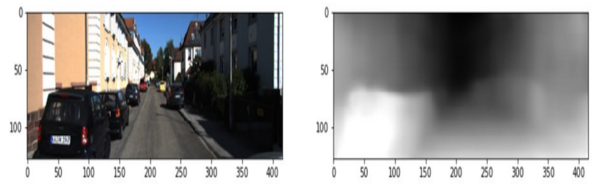
The simultaneous estimation of structure and motion is a well studied problem with established techniques its reliance on accurate image correspondence can cause problems in areas of low texture, complex geometry/photometry, thin

structures, and occlusions. Hence learning-based techniques are attractive in that they are able to leverage external supervision during training, and potentially overcome the issues when applied to test data.
One important application of geometric scene understanding is the task of novel view synthesis, where the goal is to synthesize the appearance of the scene seen from novel camera viewpoints. Recently, end-to-end learning has been applied to reconstruct novel views by transforming the input based on depth or flow, e.g., DeepStereo [11], Deep3D [20] and Appearance Flows [21]. In these methods, the underlying geometry is represented by quantized depth planes (DeepStereo), probabilistic disparity maps (Deep3D) and view-dependent flow fields (Appearance Flows), respectively. Unlike methods that directly map from input views to the target view (e.g., [22]), warping-based methods are forced to learn intermediate predictions of geometry and/or correspondence. In this work, we aim use the geometric reasoning capability from CNNs trained to perform **warping-based view synthesis**.

Our work is closely related to a line of research on learning single-view 3D inference from registered 2D observations, Garg et al. [6] propose to learn a single-view depth estimation CNN using projection errors to a calibrated stereo twin for supervision. Instead of using depth supervision, they leverage the established epipolar geometry. The color inconsistency between a left image and a synthesized left image warped from the right image is used as the supervision signal. Following this idea, Godard et al. [1] propose to constrain the left-right consistency for regularization.
In their work SfMLearner[4] , Tinghui Zhou et al., popularized the ability of deep networks for the task at hand. Their method also deals with non-rigid objects (cars, pedestrians etc.). They use an "explainability mask" in order to discount regions that violate the static scene assumption.

3. Method:





In order to infer the depth (parameterized as per-pixel depth maps under a reference view) , we first compute camera pose.

Given a pixel in an image p we can represent it by its normalized coordinates as :

$$\tilde{p} = K^{-1}\ \mathrm{p}$$

where K is the intrinsic calibration matrix of the camera. A corresponding pixel in a second view of the same scene, should lie on the corresponding epipolar line. This constraint on the pixels can be expressed using the Essential Matrix E for calibrated cameras. The Essential Matrix contains information about the relative poses between the views. Given a pixel's normalized coordinates and and that of its corresponding pixel in a second view, this relationship can be expressed as:

$$\tilde{p}^T E \mathrm{p} = 0$$

Here, E p˜ is the epipolar line in the second view corresponding to a pixel p in the first view. There could be errors in computing the pixel p, finding the corresponding pixel pˆ or in estimating the Essential Matrix E. Therefore we minimized the left hand side of the above equation by using RANSAC[27] . This is denoted as epipolar loss. By finding the Essential matrix we can then recover the pose estimate (parameterized as 6-DoF transformation matrices) that our system requires that we first train a network.

We use Nister's 5-point algorithm for estimating the Essential Matrix between two views. The solution to this gives rise to the relative poses between the cameras and the points. Absolute scale, however, cannot be recovered from just images. Nistér proposes a solution by solving a tenth degree polynomial in order to extract the Essential Matrix E which can then be decomposed into the rotation R and translation t between the two views.

We estimate dense matching points between two frames using Brute-Force Matching with SIFT Descriptors and Ratio Test or FLANN based Matcher.

3.1. Training :
Training involves a depth CNN using monocular videos, and then constraining them to predict scale-consistent results. Given two consecutive frames (I_a, I_b) sampled from an unlabeled video, we estimate their depth maps (D_a, D_b) using the ***depth CNN*** and using the relative 6D camera pose P_{ab} between them from the previous step.

With the predicted depth and relative camera pose, we can synthesize the reference image I'_a by interpolating the source image $I_{b.}$ (bilinear sampling mechanism proposed in the *spatial transformer networks* [5]).

The planar transformation that reconstructs the target view I_t by sampling pixels from a source view I_s based on the predicted depth map and the relative pose.

Image Warping:
Given a pixel p in normalized coordinates, and its depth D(p), we transform it into the source frame using the relative pose and project it onto the source image's plane are given by :

$$\hat{P} = K(\mathrm{R}_{t\to s}\ \mathrm{D(p)}\ K^{-1}\ \mathrm{p} + t_{t\to s})$$

where K is the camera calibration matrix, D(p) is the depth of pixel p, $R_{t\to s}$ and $t_{t\to s}$ are the rotation and translation respectively from the target frame to the source frame. The homogeneous coordinates of $\hat{p}$ are continuous while we require integer values. Thus, we interpolate the values from nearby pixels, using bi-linear sampling.

The depth image is generated using **novel view synthesis** using based warping as the main supervisory signal. Given the per-pixel depth and the relative pose between images, we synthesize the image of the scene from a novel viewpoint. We minimize the photometric error between the warped image and the image at the given viewpoint.

Given a target view I_t and S source views I_s , we minimize the photometric error between the target view and the source view warped into the target's frame, denoted by I^s .

$$L_{warp} = \frac{1}{N}\sum_{s=0}^{S}\sum_{p=0}^{N}|I_t(p) - \hat{I}_s(p)|$$

where N is the total number of pixels.

Loss Functions:
The network can be supervised by the ***photometric loss between the real image I_a and the synthesized one I'_a*** . However, due to dynamic scenes that violate the geometric assumption in image reconstruction (as stated in the section 3.2 below), the performance of this basic framework is limited.

Photometric Loss:
With the predicted depth map D_a and the relative camera pose P_{ab} , that we obtained from training the Depth CNN in the step above, we synthesize I'_a by warping I_b , where differentiable bilinear

interpolation is used (***inverse_warp*** in the algorithm detailed below). The Photometric Loss is given as :

$$L_p = \frac{1}{|V|}\sum_{p\in V}(\lambda_i \|I_a(p) - I'_a(p)\|_1 + \lambda_s \frac{1 - \mathrm{SSIM}_{aa'}(p)}{2}),$$

Here V stands for valid points that are successfully projected from I_a to the image plane of I_b , and |V | defines the number of points in V. $SSIM_{aa'}$ stands for the element-wise similarity between I_a and $I_{a'}$ by the SSIM function [14].
We use $\lambda_i = 0.15$ and $\lambda_s = 0.85$.

Smoothness Loss:
Issues regarding learning wrong depth values for texture-less regions, are handled by trying to ensure that the depth prediction is derived from spatially similar areas. Depth discontinuities usually occur at object boundaries hence we minimize L1 norm of the 2nd order spatial gradients of the inverse depth of a pixel weighted by the image laplacian at that pixel. This is to account for sudden changes in depth due to crossing of object boundaries and ensure a smooth change in the depth values.

$$L_{smooth} = \frac{1}{N}\sum_{p=0}^{N}\sum_{i\in\{x,y\}}\sum_{j\in\{x,y\}} |\partial_{ij} d(p)| e^{-|\partial_{ij} I(p)|}$$

where N is the total number of pixels.

Structural Similarity Loss:
A robust metric for measuring perceptual differences between two images is the Structural Similarity Index (SSIM) [42]. This is widely applied in tasks when comparing 2 images of the same scene.
SSIM considers three main factors, namely lunimance, constrast and structure, which provide a more robust measure for image similarity. Since SSIM needs to be maximized
(with 1 as the maximum value), we minimize the below loss

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

Geometry consistency loss :
We enforce geometric consistency by requiring that D_a and D_b (related by P_{ab}) conform the same 3D scene structure, and minimize their differences. This optimization not only encourages the geometry consistency between samples in a batch but also transfers the consistency to the entire sequence eg depths of I_1 agree with depths of I_2 in a batch; depths of I_2 agree with depths of I_3 in another training batch. This ensures scale-consistent predictions over the entire sequence. Hence the depth inconsistency map D_{dif}

$$D_{\mathrm{diff}}(p) = \frac{|D_b^a(p) - D'_b(p)|}{D_b^a(p) + D'_b(p)}$$

where D_b^a is the computed depth map of I_b by warping D_a using P_{ab} , and $D_b^{‘}$ is the interpolated depth map from the estimated depth map D_b .
The geometric consistency loss is defined as :

$$L_{GC} = \frac{1}{|V|}\sum_{p\in V} D_{\mathrm{diff}}(p),$$

which minimizes the geometric distance of predicted depths between each consecutive pair and enforces their scale-consistency.
The basic algorithm:
Train phase:
Let us denote < I_1 , . . . , I_N > as a training image sequence with one of the frames I_t being the target view(tgt_img) and the rest being the source views I_s (ref_imgs):

```
0. Pre-compute for
 poses,pose_inv =
compute_pose_from_image_correspondances(left_img,right_
img, K0, K1)

for all epochs:
for all target_view, source_view, camera_intrinsics in the
dataset do :
```

1. tgt_depth, ref_depths = compute_depth(tgt_img, ref_imgs) – by forward pass through depth
CNN.

2. loss1, loss3 = compute_photo_and_geometry_loss :
inverse_warp - Compute the loss based on warping nearby views to the target using the computed depth and pose.
3. loss2 = compute_smooth_loss
4. loss = w1*loss_1 + w2*loss_2 + w3*loss_3 (where w1 , w2, and w3 are hyperparameters)
5. Compute backward pass of both networks supervised by the above loss function.
6. Validate with ground truth.
7. Save the best weights.

Once the model is trained we can perform the validation and test. With the model having achieved good accuracy we proceed to inference stage.

Inference phase :

Use the saved weights to compute the depth map of a test image.

4. *Dataset and Features:*

4.1.1. The KITTI dataset [7] has been recorded from a moving platform while driving in and around Karlsruhe, Germany. It includes camera images, laser scans, high-precision GPS measurements and IMU accelerations from a combined GPS/IMU system. The main purpose of this dataset is to push forward the development of computer vision and robotic algorithms targeted to autonomous driving . This dataset can be used for evaluating state-of-the-art computer vision methods. The raw data set is divided into the categories ’Road’, ’City’, ’Residential’, ’Campus’ and ’Person’.
In the KITTI dataset most of the scenes are static without significant scene motions, and the occlusion/visibility effects only occur in small regions in sequences across a short time span (3-frames).

4.1.2. Cityscapes Dataset [8]: Large-scale dataset that contains a diverse set of stereo video sequences recorded in street scenes from 50 different cities, with high quality pixel-level annotations of 5 000 frames in addition to a larger set of 20 000 weakly annotated frames. The dataset is thus an order of magnitude larger than similar previous attempts.
The Cityscapes Dataset is intended for

4.1.2.1. Assessing the performance of vision algorithms for major tasks of semantic urban scene understanding: pixel-level, instance-level, and panoptic semantic labeling;

4.1.2.2. Supporting research that aims to exploit large volumes of (weakly) annotated data, e.g. for training deep neural networks.

5. Experiments/Results/Discussion:

We conducted training on a small dataset comprising of images from a subset of the KITTI dataset. We trained on 392 samples in 4 train scenes and the validation set consisted of 308 samples in 2 valid scenes. The training data consisted of a snippet of three sequential video frames as a training sample, where we set the second image as reference frame to compute loss with other two images and then inverse their roles to compute loss again for maximizing the data usage.

Used the ADAM [15] optimizer, and set the

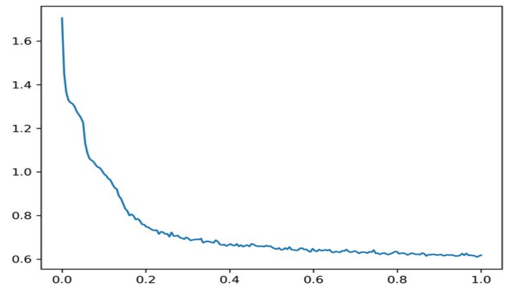
batch size = 4 and the

learning rate = 10^{-4} .

During training, used $\alpha = 1.0$, $\beta = 0.1$, and $\gamma = 0.5$.

Trained the network in 200 epochs with 1000 randomly sampled batches in one epoch, and validated the model at per epoch.

Quantitative Loss:



Training loss

Some Defined Error Metrics:

Established error metrics consider global statistics between a predicted depth map Y and its ground truth depth image Y^* with T depth pixels.

Threshold: % of y such that $\max(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}) = \sigma < thr$

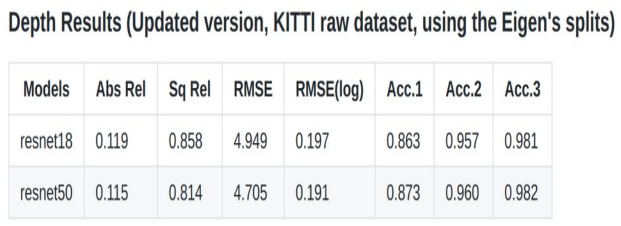
Absolute rel. diff.: $\text{rel} = \frac{1}{T}\sum_{i,j}\left|y_{i,j} - y_{i,j}^*\right| / y_{i,j}^*$

Squared rel. diff.: $\text{srel} = \frac{1}{T}\sum_{i,j}\left|y_{i,j} - y_{i,j}^*\right|^2 / y_{i,j}^*$

RMS (linear): $\text{RMS} = \sqrt{\frac{1}{T}\sum_{i,j}\left|y_{i,j} - y_{i,j}^*\right|^2}$

RMS (log): $\log_{10} = \sqrt{\frac{1}{T}\sum_{i,j}\left|\log y_{i,j} - \log y_{i,j}^*\right|^2}$

Error Metrics:Results of the Baseline implementation as detailed in [12] :



Depth Results (Updated version, KITTI raw dataset, using the Eigen's splits)

Models	Abs Rel	Sq Rel	RMSE	RMSE(log)	Acc.1	Acc.2	Acc.3
resnet18	0.119	0.858	4.949	0.197	0.863	0.957	0.981
resnet50	0.115	0.814	4.705	0.191	0.873	0.960	0.982

Our Error Metrics:

Avg abs_diff : 3.375,

abs_rel : 0.214,

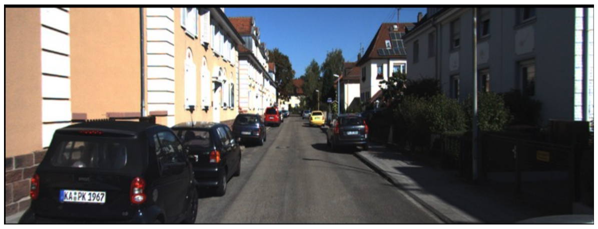
sq_rel : 1.681,

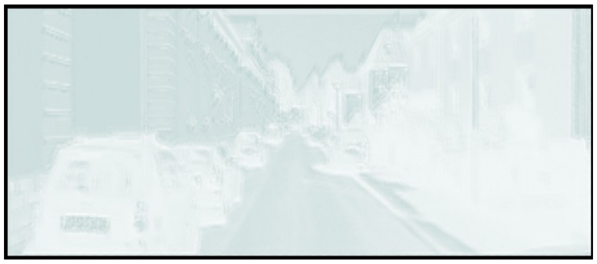
a1 : 0.694,

a2 : 0.887,

a3 : 0.955

Qualitative Results:





Network Architecture:

Our network architecture comprises of
Encoder Network:

The encoder extracts high level and low-resolution features from the input image. These are then fed to the decoder, where they are gradually up-sampled, refined, and merged to form the final high-resolution output depth map. Our design choice for the encoder was guided by the fact that we would like this to run on mobile devices for inference – hence we opted for low latency designs.

Encoders used in depth estimation networks are commonly those that are used for image classification. Popular choices include VGG-16 [17] and ResNet-50 [18] because of their strong expressive power and high accuracy. However, such networks are have high complexity and latency, making them unsuitable for applications running mobile devices[16].

We based our encoder on MobileNet [19]. The encoder makes use of depth-wise decomposition, factorizing an m×m×n standard convolutional layer into n m×m depth-wise layers and a 1×1 pointwise layer.
Since each filter in a depth-wise layer only convolves with a single input channel, the complexity of a depth-wise layer is much lower than that of a standard convolutional layer, where each filter convolves with all input channels. Also each pointwise filter is just a 1×1 kernel, so the number of multiply-accumulate operations performed by a point-wise layer is much smaller than that of the original standard convolution. Hence, depth-wise decomposition significantly reduces the complexity of a convolutional layer, making MobileNet more efficient than networks with standard convolution like ResNet and VGG and this translates to reduced latency.

Decoder Network: The decoder merges and up-samples the output of the encoder to form a dense prediction. The key design aspect here is the up-sample operation used (e.g., unpooling, transpose convolution, interpolation combined with convolution).

The decoder that we used consists of five up-sample layers and a single pointwise layer at the end. Each up-sample layer performs convolution and reduces the number of output channels by 1/2 relative to the number of input channels. This is followed by nearest-neighbor interpolation that doubles the spatial resolution of intermediate feature maps. Depth-wise decomposition to further lower the complexity of all convolutional layers, resulting in slim and fast decoder. The network complexity can be measured by indirect metrics such as multiply-accumulate operations and latency on a target hardware platform.

6. Conclusion:

In this work I have presented an end-to-end learning pipeline that utilizes the task of view synthesis for supervision of single-view depth estimation. While the supervisory task of training the network to estimate depth show satisfactory results – future enhancements could involve using a method where disparity values could be regressed from a cost volume. Also the network architecture could be enhanced using skip connections. Encoder networks typically contain many layers to gradually reduce the spatial resolution and extract higher-level features from the input. The output of the encoder into the decoder becomes a set of low resolution features in which many image details can be lost, making it more difficult for the decoder to recover pixel-wise (dense) data. Skip connections allow image details from high resolution feature maps in the encoder to be merged into features within the decoder; this helps the decoding layers reconstruct a more detailed dense output.

7. References:

[1] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In Computer Vision and Pattern Recognition, 2017.

[2] Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003.

[3] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In Neural Information Processing Systems (NIPS), 2014.

[4] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[5] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In Advances in Neural Information Processing Systems, pages 2017–2025, 2015.

[6] R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In European Conf. Computer Vision, 2016.

[7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite.

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler,R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3213–3223, 2016.

[9] 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.

[10] C. Fehn. Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv. In Electronic Imaging 2004, pages 93–104. International Society for Optics and Photonics, 2004.

[11] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deep-Stereo: Learning to predict new views from the world's imagery. In Computer Vision and Pattern Recognition, 2016.

[12] Bian, Jia-Wang and Li, Zhichao and Wang, Naiyan and Zhan, Huangying and Shen, Chunhua and Cheng, Ming-Ming and Reid, Ian. Unsupervised Scale-consistent Depth and Ego-motion Learning from Monocular Video. In Thirty-third Conference on Neural Information Processing Systems (NeurIPS), 2019.

[13] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. *TensorFlow: A System for Large-Scale Machine Learning*. In OSDI.

[14] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image Quality Assess-ment: from error visibility to structural similarity. IEEE Transactions on Image Processing (TIP), 13(4), 2004.

[15] Diederik P Kingma and Jimmy Ba. ADAM: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[16] Wofk, Diana and Ma, Fangchang and Yang, Tien-Ju and Karaman, Sertac and Sze, Vivienne. FastDepth: Fast Monocular Depth Estimation on Embedded Systems. IEEE International Conference on Robotics and Automation (ICRA), 2019

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition, International Conference on Learning Representations (ICLR), 2015.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[20] J. Xie, R. B. Girshick, and A. Farhadi. Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks. In European Conf. Computer Vision, 2016.

[21] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In European Conference on Computer Vision, pages 286–301. Springer, 2016.

[22] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. In European Conference on Computer Vision, pages 322–337. Springer, 2016.

[23]. Xingkui Wei and Yinda Zhang and Zhuwen Li and Yanwei Fu and Xiangyang Xue. DeepSFM: Structure From Motion Via Deep Bundle Adjustment. ECCV 2020

[24]. Wang, Jianyuan and Zhong, Yiran and Dai, Yuchao and Birchfield, Stan and Zhang, Kaihao and Smolyanskiy, Nikolai and Li, Hongdong. Deep Two-View Structure-From-Motion Revisited. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8953-8962,2021

[25]. Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. Communications of the ACM, 54(10):105–112, 2011.

[26]. H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. Nature, 293(5828):133–135, 1981

[27] Martin A Fischler and Robert C Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM (1981).