POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Trifocal Tensor Estimation for n-view Deep Structure-from-Motion

Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica

Author: Leonardo Perelli

Advisor: Prof. Luca Magri

Co-advisor: Dr. Andrea Porfiri dal Cin, Prof. Giacomo Boracchi

Academic year: 2021-2022

1. Introduction

In this work we address the problem of Structure-from-Motion (*SfM*), a fundamental task in computer vision with applications spanning from autonomous driving to augmented reality. The SfM problem consists in recovering the 3D structure of a scene starting from n images taken from different viewpoints. The 3D structure of a scene can be recovered by estimating the position of the cameras from which images were taken (*camera pose*) as well as the depth measurements at each pixel of the images (*depth maps*). The depth map for a given *reference* image is estimated using neighbouring *target* images. An important parameter that determines the quality of the 3D reconstruction is the number n of images (or *views*) used to produce a single depth map. In the base case $n = 2$, one target image is used to predict the depth map of the reference image. To leverage multiple images, it is required to estimate the camera pose between the reference frame and all target frames. The typical SfM pipeline proceeds in two main stages: (i) recover camera pose (ii) use the estimated camera pose to compute depth maps. In this thesis, we focus on improving the camera pose estimates. Indeed, the majority of deep learning SfM pipelines directly regress the camera pose, providing the images as input to a neural network. While these pipelines can handle the general n-view case, they do not exploit the constraints of 3D geometry, over-relying on neural networks. Some pipelines try to refine the noisy camera poses obtained with neural networks by introducing additional computation in the form of repeated iterations of the algorithms, which is very computationally expensive. To overcome these limitations, the pipeline introduced in [5] leverages the 3D geometric knowledge and couples it with the use of neural networks. However, this approach is limited to the case $n = 2$. In this thesis, our contribution is two-fold. Inspired by [5], we propose an SfM pipeline for the general n-view case which efficiently couples neural networks with the use of 3D geometric constraints. The pipeline leverages the Trifocal tensor and proposes a novel pose chaining algorithm to expand camera pose estimation to the general case of n images. We also provide a comparison between different Trifocal tensor estimation algorithms together with their implementation.

2. Problem Formulation

The objective of SfM pipelines is to provide a 3D reconstruction of the scene. To do this,

the pipelines are fed as input n 2D images in RGB format $\{\mathrm{I}_i\}_{i=1}^n$ such that $\mathrm{I}_i \in \mathbb{R}^{\mathrm{H}\times\mathrm{W}\times 3}$ $\forall i = 1, ..., n$. The pipelines also receive the intrinsic parameters $\{\mathrm{K}_i\}_{i=1}^n$ of the cameras from which $\{\mathrm{I}_i\}_{i=1}^n$ are taken, where $\mathrm{K}_i \in \mathbb{R}^{3\times 3}$ $\forall i = 1, ..., n$. The outputs of the pipelines are a pair $(\mathrm{R}_i, \mathbf{t}_i)$ $\forall\ i = 1, ..., n$ and a depth map $\mathrm{D}_i \in [\mathrm{d}_{min}, \mathrm{d}_{max}]^{\mathrm{H}\times\mathrm{W}}$ $\forall\, i = 1, ..., n$. $\mathrm{R}_i \in SO(3)$ represents the camera rotation and $\mathbf{t}_i \in \mathbb{R}^3$ is the translation vector of the camera centre. D_i assigns to every pixel in the image a depth value, which represents the distance from the camera to the object at that pixel location.
The quality of D_i is evaluated with respect to the ground truth depth map $\mathrm{D}_{gt,i}$ through common metrics such as the*Absolute relative difference*, the *Squared relative difference*, the *RMSE* and $RMSE_{log}$. They can be computed as:

$$\text{Absolute rel. diff.} = \frac{1}{n}\sum_{i=1}^{n}\|\mathcal{D}_i - \mathrm{D}_{i,gt}\|_1/\mathrm{D}_{i,gt}$$

$$\text{Square rel. diff.} = \frac{1}{n}\sum_{i=1}^{n}\|\mathrm{D}_i - \mathrm{D}_{i,gt}\|_2^2/\mathrm{D}_{i,gt}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\|\mathrm{D}_i - \mathrm{D}_{i,gt}\|_2^2}$$

$$\text{RMSE}_{log} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\|log(\mathrm{D}_i) - log(\mathrm{D}_{i,gt})\|_2^2}$$

SfM pipelines make several assumptions. The scenes are required to be static, i.e. they contain no moving objects. The lighting conditions in the scene should be consistent across all I_i. Sufficient texture in the scene is required: images with extended textureless regions make it much more difficult to match points over different images. Finally, SfM pipelines require some overlap between the images as well as sufficiently textured regions.
Unfortunately, it is common to have scenes containing dynamic objects which can perturb both the camera pose and depth map estimation, hence SfM pipelines are required to be robust with respect to these conditions.

3. Related Work

We now explore recent pipelines proposed in the literature. Figure 1 provides an overview of the main characteristics of the pipelines.

	Limited use of geometric knowledge	Efficient use of geometric knowledge
2-view	DeMoN [4]	2-ViewRevisited [5]
n-view	MVSNet [7] DPSNet [2] DeepSfM [6]	?

Figure 1: Taxonomy of pipelines. Most of the pipelines make limited or no use of the epipolar constraints. 2-ViewRevisited [5] couples neural networks with the epipolar constraints, but is limited to the 2-view case. No such pipelines tackle the n-view setting.

3.1. 2 view Deep pose estimation

Camera pose estimation, as depth map estimation, can benefit from the use of neural networks. Since 3D geometry is well studied, it is important to leverage this knowledge as much as possible, using neural networks only in tasks they can excel at. DeMoN [4] is one of the first approaches to formulate SfM as a learning problem in the domain of neural networks. The main contribution of the approach lies in the introduction of an end-to-end differentiable neural network architecture to estimate camera pose and depth maps. The problem is formulated as a regression: the network is fed $n = 2$ images I_a, I_b and has to recover D_i, R_i, $\mathbf{t}_i$. The approach has two main limitations: (i) it has to learn the real scale of the scene from the images, which is an ill-posed problem that can cause the network to overfit and (ii) it does not incorporate knowledge of epipolar geometry constraints, which instead have to be learned from scratch and could limit ability to generalise.
To overcome these limitations, [5] couples neural networks with traditional geometric tools. It uses neural networks to estimate optical flow O_{ab} between I_a and I_b. Optical flow is the task of matching points between two different images. Then, it uses the point matches provided by O_{ab} to compute the Fundamental matrix F, which encodes the epipolar geometry between I_a and I_b. From the Fundamental matrix, camera pose can be easily recovered. This is the standard approach in traditional SfM pipelines, but neural networks can now improve the optical flow estimation. The advantages of this pipeline are several: (i) it efficiently couples neural networks together with 3D geometrical constraints and (ii)

the camera pose computed through the Fundamental matrix is estimated in a normalized scale rather than the real scale, so the camera pose estimation is well-posed.

3.2. n-view

Different algorithms have been proposed to deal with the n-view case. We first discuss the depth estimation step, which is shared by most works such as [6][2][7]. Later, we will discuss the camera pose estimation step.
During the depth estimation step, the plane sweep algorithm is used to generate a cost volume and regress a depth map. Plane sweep is an algorithm used to estimate the most likely depth value for a specific pixel $\mathbf{x} \in \mathrm{I}_a$ by sweeping through S depth samples $\{d_j\}_{j=1}^{S}$ in a range $[d_{min}, d_{max}]$. The pixel is projected from I_a to the corresponding $\tilde{\mathbf{x}} \in \mathrm{I}_b$ assuming it has depth d_j following:

$$\tilde{\mathbf{x}} \sim \mathrm{K}\,[\mathrm{R} \mid \mathbf{t}] \begin{bmatrix} \left(\mathrm{K}^{-1}u\right) d_j \\ 1 \end{bmatrix} \tag{1}$$

A cost is assigned based on the similarity between the $\mathrm{F}_a(\mathbf{x})$ and $\mathrm{F}_b(\tilde{\mathbf{x}})$, where $\mathrm{F}_a(\mathbf{x})$ and $\mathrm{F}_b(\tilde{\mathbf{x}})$ are the feature maps extracted from I_a and I_b. This is repeated $\forall j = 1, ..., \mathrm{S}$ and $\forall\ \mathbf{x} \in \mathrm{I}_a$, leading to a 4D cost volume $\mathrm{V} \in \mathbb{R}^{H \times W \times Ch \times S}$, where Ch is the number of channels in the feature map. The 4D cost volume then goes through several convolutional layers that reduce it to a 3D volume $\mathrm{V} \in \mathbb{R}^{H \times W \times S}$. $\mathrm{V}(\mathbf{x})$ represents a probability distribution over the depth samples and is used to regress a depth map. The advantage of this approach is that (i) it injects 3D knowledge in the network, as the pixel projection follows the epipolar constraints and (ii) the n-view case can be tackled by aggregating multiple cost volumes, each produced by comparing the reference image and the target images. The main requirement of the plane sweep algorithm is to have camera poses between reference and target images all expressed in the same scale. This is required to keep the same proportions of the real scene without any distortion. For example, [4] produces $\{\mathrm{R}_1, \mathrm{R}_2, \mathbf{t}_1, \mathbf{t}_2\}$ in real scale (meters).
During the camera pose estimation step, [6] computes pairwise camera poses with [4] and uses them as initialization. The authors then refine the inaccurate $\{\mathrm{R}_i, \mathbf{t}_i\}_{i=1}^{n}$ by performing multiple iterations of a plane sweep algorithm adapted to produce camera pose samples rather than depth samples. This injects some geometrical knowledge in the network, but using [4] as initialization still carries the limitations discussed in 3.1. Overall, the use of cost volumes in camera pose estimation is expensive and the iterative use of plane sweep exacerbates the problem.
Expanding the pipeline of [5] to the n-view case is not as easy. Indeed, the camera pose is normalised and all scale information is lost. Therefore, it is not possible to leverage pairwise camera pose estimates in the same way as [6] leverages [4].

4. Proposed Solution

The core of our proposal is to use the Trifocal tensor to extend the pipeline proposed by [5] to the general n-view case. The main idea behind the pipeline is to compute the Trifocal tensor for multiple triplets of overlapping cameras and rescale all the camera pose estimates to share the same scale. Figure 2 illustrates our proposed pipeline.

4.1. Pipeline

The pipeline is composed of three main steps, namely (i) 3-view point matching, (ii) camera pose estimation and (iii) depth map estimation. The 3-view point matching starts off by selecting $n-2$ overlapping triplets which are used to estimate $[\mathrm{R}_i|\mathbf{t}_i]_{i=1}^{n}$. For every triplet $(\mathrm{I}_a, \mathrm{I}_b, \mathrm{I}_c)$, we recover 3-view point matches $\mathbf{x} \leftrightarrow \mathbf{x}' \leftrightarrow \mathbf{x}''$ by estimating the optical flows O_{ab} and O_{bc} between the couples $(\mathrm{I}_a, \mathrm{I}_b)$ and $(\mathrm{I}_b, \mathrm{I}_c)$. The point matches obtained from O_{ab} and O_{bc} are then intersected to produce the 3-view point matches $\mathbf{x} \leftrightarrow \mathbf{x}' \leftrightarrow \mathbf{x}''$.
Using the 3-view point matches $\mathbf{x} \leftrightarrow \mathbf{x}' \leftrightarrow \mathbf{x}''$ we estimate the Trifocal tensor T. The camera poses $[\mathrm{R}_{ba}|\mathbf{t}_{ba}]$ and $[\mathrm{R}_{bc}|\mathbf{t}_{bc}]$ are extracted from T. We compared different algorithms to estimate the Trifocal tensor. The algorithms were proposed in [1] and [3]. We then rescale and chain the camera pose estimates obtained in different triplets to recover the n-view normalised camera poses $[\mathrm{R}_i|\mathbf{t}_i]_{i=1}^{n}$ with a shared scale.
Finally, the plane sweep algorithm is carried out by using the normalised camera poses $[\mathrm{R}_i|\mathbf{t}_i]_{i=1}^{n}$. Even though the camera configuration is nor-

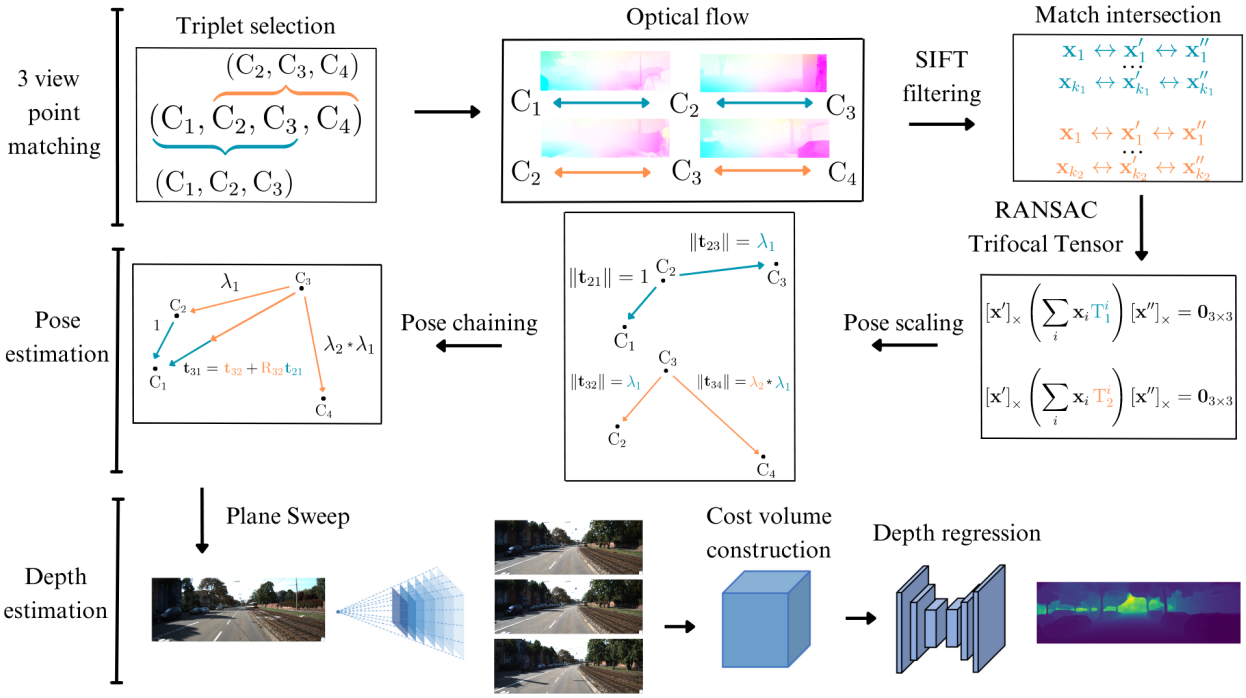

Figure 2: Our proposed pipeline when $n = 4$. The pipeline is a sequence of three main steps: 3-view point matching, pose estimation and depth estimation.

malised, it preserves the original geometry of the scene and the plane sweep algorithms outputs a normalised depth map D for the reference frame I_{ref}.

4.2. Pose scaling and chaining

As previously mentioned, the camera poses $[\mathrm{R}_i|\mathbf{t}_i]$ extracted from T_i are expressed in different scale for every i-th triplet. To perform the n-view plane sweep algorithm however, the camera poses have to share a common scale. If this condition is not met, the 3D geometry of the scene is distorted. Our proposal is to rescale all camera poses $[\mathrm{R}_i|\mathbf{t}_i]$ with respect to the scale of $\mathbf{t}_1$. We provide an example in the case $n = 4$. Given two overlapping triplets $(\mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3)$ and $(\mathrm{C}_2, \mathrm{C}_3, \mathrm{C}_4)$, we rescale translation vectors from $(\mathrm{C}_2, \mathrm{C}_3, \mathrm{C}_4)$ to match the scale of $(\mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3)$. Let $\mathbf{t}_{23}^{1,2,3}$ be the translation between C_2 and C_3 computed in triplet $(\mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3)$ and $\mathbf{t}_{32}^{2,3,4}$ the translation between C_3 and C_2 computed in triplet $(\mathrm{C}_2, \mathrm{C}_3, \mathrm{C}_4)$. Since we have estimated the translation vector $\mathbf{t}_{23}$ in two different triplets, we can match the scale of $\mathbf{t}_{32}^{2,3,4}$ to that of $\mathbf{t}_{23}^{1,2,3}$. Indeed,

$$\lambda_1 \mathbf{t}_{32}^{2,3,4} \approx \mathbf{t}_{32}^{1,2,3}$$

since

$$\|\mathbf{t}_{23}^{1,2,3}\| = \lambda_1$$
$$\|\mathbf{t}_{32}^{2,3,4}\| = 1$$

Therefore, we can conclude that $\lambda_1 \mathbf{t}_{32}^{2,3,4}$ and $\lambda_1 \mathbf{t}_{34}^{2,3,4}$ match the scale of the previous triplet. As a result, after dropping the triplet notation, the camera poses $[\mathrm{R}_{21}|\mathbf{t}_{21}], [\mathrm{R}_{23}|\mathbf{t}_{23}], [\mathrm{R}_{34}|\lambda_1 \mathbf{t}_{34}]$ share a common scale that preserves the proportions induced by the real scene. This is achieved even if the scale does not match the ground truth scale (e.g. meters).

The camera poses computed in step 2.2 are a sequence of relative poses between neighbouring cameras. The plane sweep algorithm requires the camera pose between the reference frame I_{ref} and each target frame $\{\mathrm{I}_{tar_i}\}_{i=1}^{n-1}$. Let image I_3 be the reference frame. Then, the plane sweep requires $[\mathrm{R}_{31}|\mathbf{t}_{31}], [\mathrm{R}_{32}|\mathbf{t}_{32}], [\mathrm{R}_{34}|\lambda_1 \mathbf{t}_{34}]$. We are thus missing: $[\mathrm{R}_{31}|\mathbf{t}_{31}]$ and $[\mathrm{R}_{32}|\mathbf{t}_{32}]$.

We can recover $[\mathrm{R}_{32}|\mathbf{t}_{32}]$ by inverting the pose $[\mathrm{R}_{23}|\mathbf{t}_{23}]$, which we already computed.

Similarly, we can also recover $[\mathrm{R}_{31}|\mathbf{t}_{31}]$. This can be achieved by chaining available poses. In this case, we can chain the pose $[\mathrm{R}_{32}|\mathbf{t}_{32}]$ to $[\mathrm{R}_{21}|\mathbf{t}_{21}]$. As before, $[\mathrm{R}_{32}|\mathbf{t}_{32}]$ is obtained by inverting $[\mathrm{R}_{23}|\mathbf{t}_{23}]$.

Pipeline	*lower is better*				*higher is better*		
	Abs Rel	Sq Rel	RMSE	RMSE_{log}	α^1	α^2	α^3
2-ViewRevisited [5]	**0.055**	**0.224**	2.273	**0.091**	**0.956**	**0.984**	**0.993**
Ours, $n = 3$	0.056	0.261	**2.264**	0.095	0.951	0.983	0.992
Ours, $n = 4$	0.090	0.419	2.670	0.128	0.899	0.968	0.987

Table 1: Results on KITTI Depth Eigen split. Incresing n leads to worse results due to dynamic objects.

5. Experimental evaluation

We now report the results obtained on different benchmarks. We only report the results obtained by our full pipeline on the depth estimation task, while we omit the experiments used to choose the best Trifocal tensor estimation algorithm.

5.1. Datasets

We mainly focus on 2 datasets to validate the performance of our depth estimation algorithm. In particular, we run our experiments on KITTI Depth and ETH3D test sets, which are also used by [5] and [6]. The KITTI Depth dataset is mainly designed to evaluate the performance of monocular depth estimation algorithms in self-driving cars and provides two splits. The Eigen split, which consists of 697 individual frames, is the most common amongst the KITTI splits. [5] introduced the Eigen SfM split, a subset with 256 frames filtered to exclude scenes containing dynamic objects or near static motion. The ETH3D dataset contains 454 frames over 13 different scenes. The dataset features both indoor and outdoor scenes collected with the aid of very accurate laser measurements. As such, the 3D groundtruth points are reliable and account for a very solid benchmark. Since our network has only been trained on the KITTI dataset, we expect our pipeline to under-perform competitors such as [6] on the ETH3D, since they have trained the network on datasets similar to ETH3D.

5.2. KITTI Depth

We run our experiments on both the splits of the KITTI dataset. Table 1 reports the results on the Eigen split. From the table, it appears that increasing the number of views has barely no effect when $n = 3$ and even worsens performance when $n = 4$. However, inspecting the depth maps obtained one can notice that the static parts of the scene are estimated better. That is, the noise and depth artifacts inside the estimated depth maps is strongly reduced. The reason why numerical results do not improve can be traced back to the presence of dynamic objects. Indeed, using more frames to estimate the depth map of the reference image implies that dynamic objects will have travelled greater distances between the reference frame and target frame. This happens because the KITTI Depth dataset is a video sequence, hence distant frames have greater time spans between them. Moreover, objects near the edge of the image can easily disappear in later frames. Hence, when computing the cost volumes through plane sweep, points are not matched correctly: the warping accounts for the camera motion but not for the object motion.
With this in mind, we test the algorithm on the KITTI Depth Eigen SfM split, which should contain less dynamic objects. The results obtained in Table 2 this time show a very good performance of the algorithm, which improves by around 18% to 47% over an already state-of-the-art method. The 3-view algorithm is already better than the original [5], and the 4-view manages to improve slightly more.

5.3. ETH3D

We proceed to test our pipeline on the ETH3D dataset. We perform the experiments both estimating the camera pose with the Trifocal tensor, both using the real ground truth camera poses. In all cases, we can see that increasing n has an important effect on the performance of our pipeline. The effect seems particularly strong when the camera pose estimates are noisy. In such case, our pipeline provides an improvement over [5] which ranges from 49% to 150%. When using the ground-truth camera poses instead, the improvement of $n > 3$ views over the 2-view

Pipeline	*lower is better*				*higher is better*		
	Abs Rel	Sq Rel	RMSE	RMSE_{log}	α^1	α^2	α^3
2-ViewRevisited [5]	0.034	0.103	1.919	0.057	0.989	0.998	0.999
Ours, $n = 3$	**0.027**	0.071	1.617	**0.048**	**0.992**	0.998	0.999
Ours, $n = 4$	0.028	**0.070**	1.599	0.049	0.992	0.998	0.999
Ours, $n = 5$	0.031	0.073	**1.593**	0.052	0.991	0.998	0.999

Table 2: Results on the KITTI Depth Eigen SfM split. Increasing n yields benefits ranging from 18% to 47 %.

pipeline from [5] is reduced to a smaller range of 6% to 15%. Of course, as our pipeline has not been explicitly fine-tuned for this dataset, it would be best to run these experiments with the fine-tuned pipeline and check that the obtained results still hold. The performance improves when accounting for more views, but it still lags other fine-tuned SfM algorithms such as [6].

6. Conclusion

In this work we proposed an SfM pipeline which handles the general n-view case while solving a well posed problem and exploiting well know 3D constraints. The provided experimental evaluation showed that using higher values of n in presence of dynamic objects can lead to worse depth map estimates due to dynamic objects. However, we showed significant improvements over 2-view state-of-the-art pipelines when the scene is static. In particular, the performance of our pipeline on the KITTI Eigen SfM split displays improved performance when setting $n \geq 3$. Our tests on ETH3D also provided promising results, showing that increasing n has a positive effect, especially when the camera pose estimate is noisy. Interesting directions for future research include accounting for the motion of dynamic objects in the plane sweep algorithm and further exploiting loop constraints during the camera pose estimation step.

References

[1] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, 2 edition, 2004.

[2] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. DPSNet: End-to-end Deep Plane Sweep Stereo, May 2019. arXiv:1905.00538 [cs].

[3] Laura F. Julià and Pascal Monasse. A Critical Review of the Trifocal Tensor Estimation. In Manoranjan Paul, Carlos Hitoshi, and Qingming Huang, editors, *Image and Video Technology*, Lecture Notes in Computer Science, pages 337–349, Cham, 2018. Springer International Publishing.

[4] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. DeMoN: Depth and Motion Network for Learning Monocular Stereo. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5622–5631, July 2017. arXiv:1612.02401 [cs].

[5] Jianyuan Wang, Yiran Zhong, Yuchao Dai, Stan Birchfield, Kaihao Zhang, Nikolai Smolyanskiy, and Hongdong Li. Deep Two-View Structure-from-Motion Revisited, April 2021. arXiv:2104.00556 [cs].

[6] Xingkui Wei, Yinda Zhang, Zhuwen Li, Yanwei Fu, and Xiangyang Xue. DeepSFM: Structure From Motion Via Deep Bundle Adjustment, August 2020. arXiv:1912.09697 [cs].

[7] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth Inference for Unstructured Multi-view Stereo, July 2018. arXiv:1804.02505 [cs].