Multi-body Depth and Camera Pose Estimation from Multiple Views - Supplementary Materials -

Andrea Porfiri Dal Cin
Politecnico di Milano
andrea.porfiridalcin@polimi.it

Giacomo Boracchi
Politecnico di Milano
giacomo.boracchi@polimi.it

Luca Magri
Politecnico di Milano
luca.magri@polimi.it

This document provides additional materials omitted from the main manuscript due to space restrictions. First, we provide further insights on our Multi-Body Plane Sweep Network. Specifically, we provide an in-depth look at the depth cost volume construction and regularization (Sec. 1). Then, we describe the algorithmic and implementation details of our framework (Sec. 2) and discuss the main parameters of our solution. In Sec. 3, we provide a description of the proposed *Multi-body Unstructured* dataset used in conjunction with the ETH3D and KITTI Depth datasets for multi-body evaluation. In Sec. 4, we describe the metrics used in the main manuscript for depth evaluation. In Sec. 5, we compare our method against traditional SfM baselines on static datasets and present additional experiments to evaluate the behavior of our method when it operates outside its optimal conditions. Finally, in Sec. 6, we illustrate our implementation and list third-party software used in the development process.

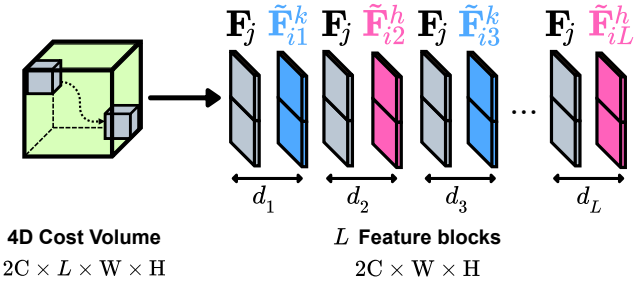

Figure 1: **Cost volume construction by multi-body plane sweep.** For each virtual depth plane at $d_l \in \{d_l\}_{l=1}^{L}$, we generate a 2C × W × H feature map by concatenating the target feature $\mathbf{F}_j$ (gray) to the warped source feature $\tilde{\mathbf{F}}_{il}^{\gamma}$ (in color) at depth d_l using motion $[\hat{\mathbf{R}}_{\gamma}^{\alpha}, \hat{\mathbf{t}}_{\gamma}^{\alpha}]$. The 4D cost volume is constructed through concatenation of these L feature blocks. $\tilde{\mathbf{F}}_{il}^{k}$ (in blue) indicates a source feature $\mathbf{F}_i$ warped onto $\mathbf{F}_j$ through the homography induced by the generic plane at d_l and its relative camera motion $[\hat{\mathbf{R}}_{k}^{\alpha}, \hat{\mathbf{t}}_{k}^{\alpha}]$. Instead, $\tilde{\mathbf{F}}_{il}^{h}$ is induced by the motion $[\hat{\mathbf{R}}_{h}^{\alpha}, \hat{\mathbf{t}}_{h}^{\alpha}]$ (in pink).

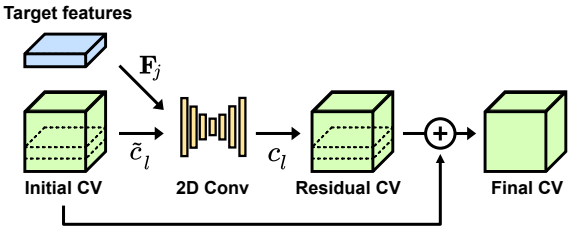

Figure 2: **Context Network for Edge-Preserving Filtering.** Each slice c_l of the regularized 3D cost volume of size $L \times \mathrm{W} \times \mathrm{H}$ and the target feature $\mathbf{F}_j$ are the input of a context network that produces a residual cost volume. The final cost volume is obtained by adding the initial and residual cost volumes.

1. Depth Cost Volume

We discuss the construction and regularization of the cost volume used in the multi-body depth estimation branch of our multi-body plane sweep network.

1.1. Construction

In this section, we provide a visual understanding of how the 4D depth cost volume is built in our multi-body depth estimation branch from an image pair $\alpha = (\mathbf{I}_i, \mathbf{I}_j)$ involving two rigidly moving objects β_k and β_h. As shown in Fig. 1, for each virtual plane at depth d_l, we concatenate the target image feature $\mathbf{F}_j$ and the source feature $\tilde{\mathbf{F}}_{il}^{\gamma}$ warped through the homography induced by the plane at d_l and its associated camera motion $[\hat{\mathbf{R}}_{\gamma}^{\alpha}, \hat{\mathbf{t}}_{\gamma}^{\alpha}]$ referring to the generic object β_γ. In Fig. 1, the motions $[\hat{\mathbf{R}}_{k}^{\alpha}, \hat{\mathbf{t}}_{k}^{\alpha}]$ (in blue) and $[\hat{\mathbf{R}}_{h}^{\alpha}, \hat{\mathbf{t}}_{h}^{\alpha}]$ (in pink) are assigned cyclically to each of the L depth planes, meaning that $d_1, d_3, d_5, \ldots$ are assigned to the k-th motion and $d_2, d_4, d_6, \ldots$ are assigned to the h-th motion. As each CNN feature is of size C × W × H, by concatenating the features along the first dimension (C), we obtain a total of L feature blocks of size 2C × W × H. Finally, we pack the L concatenated feature blocks and arrange them into a 4D cost volume of size $2\mathrm{C} \times L \times \mathrm{W} \times \mathrm{H}$, as shown in Fig. 1.

	$L = 64$			$L = 96$		
	L1-inv	Sc-inv	L1-rel	L1-inv	Sc-inv	L1-rel
all frames	0.178	0.194	0.179	0.167	0.190	0.164
2 motions only	0.162	0.181	0.152	0.158	0.179	0.148
3 motions only	0.201	0.212	0.192	0.182	0.201	0.179

Table 1: **Depth evaluation in Multi-body Unstructured.** We consider a varying number of motions M and of virtual planes L. For 2 motions only we observe a mean performance up-lift of 2.11% across all metrics going from $L = 64$ to $L = 96$, whereas for 3 motions only we observe a mean performance up-lift of 7.72%. By using 4 input frames from consecutive sequences, our method attains 13.1% better depth estimation performance, whereas [16] improves by 12.4%.

1.2. Regularization

As described, in Sec. 4.3.1 of the main manuscript, regularizing the cost volume is fundamental to cope with imperfect latent feature matching, which is especially common in image regions with uniform pixel intensity. In this section, we provide further details about our regularization strategy and, specifically, we describe the context network used to perform edge-preserving filtering, whose main steps are illustrated in Fig. 2. As described in the main manuscript, we obtain an initial 3D cost volume (Initial CV) of dimension $L \times \mathrm{W} \times \mathrm{H}$ by applying a sequence of 3D convolutions to the 4D cost volume of dimension $2\mathrm{C} \times L \times \mathrm{W} \times \mathrm{H}$. Then, we apply edge-preserving filtering to the initial 3D cost volume by employing a context network inspired by [9] to improve depth estimation. The context network takes as input a single slice $\tilde{c}_l$ of the initial 3D cost volume and the target image feature $\mathbf{F}_j$ and returns the corresponding refined cost volume slice c_l. The edge-preserving filtering step is performed for each cost slice $\tilde{c}_l, l \in 1 \dots L$ to obtain a residual cost volume (Residual CV) that is then added to the initial cost volume to obtain the final cost volume (Final CV). All cost volume slices $\tilde{c}_l$ are processed with shared weights of the context network. The context network consists of seven convolutional layers with 3×3 filters, each with a different receptive field (1,2,3,8,16,1, and 1).

The cost volume is finally upsampled to the original size of the RGB images, which have dimension $3 \times 4\mathrm{W} \times 4\mathrm{H}$, via bilinear interpolation to then regress pixel-wise depth maps of the target image $\mathbf{I}_j$.

2. Algorithmic and Implementation Details

In this section, we report details regarding the deep neural networks used in our implementation. Hyper-parameters and training details are also specified. We also report the required training time on our system.

2.1. Hyperparameters: L and M

In this section, we provide an alternative visualization of the results reported in Tab. 3 of the main manuscript to support the claim made in Sec. 5.4 that the number of motions M and the number of virtual 3D depth planes L have a significant impact on depth estimation accuracy. In Tab. 1, we report results obtained by our method on the *Multi-body Unstructured* dataset by varying the parameters of our depth estimation module. Specifically, we vary the number L of virtual 3D depth planes and consider $L = 64$ and $L = 96$, with the latter used in our final implementation. We evaluate our method for varying values of L by considering 3 different configurations: *i*) all frames in the dataset, *ii*) frames with 2 motions exactly, *iii*) frames with 3 motions exactly. Results show that increasing the number of depth planes L is more beneficial when considering only frames with 3 motions, whereas, in case of 2 motions, we observed more modest improvements in accuracy. When considering 2 motions only, we observe a mean performance uplift on the depth evaluation metrics of 2.11%, whereas when considering 3 motions the improvement is more substantial at 7.72%. We attribute these differences to the way the 4D depth cost volume is built in plane sweep depth estimation architectures. Our multi-body plane sweep assigns a maximum number of L/M depth planes to each rigid motion in the scene for feature warping. As the number of motions increases, the fewer depth planes are assigned to each rigid motion, resulting in a direct impact in accuracy as shown in Tab. 1.

2.2. Optical Flow Fine-Tuning

For all experiments, the optical flow is trained on synthetic scenes from [15], except for KITTI Depth, where, for a fair comparison, we follow the same fine-tuning procedure as in Wang et al. [14]. Other scenes do not require fine-tuning, nor additional computational and labelling costs, as our joint depth and pose network can refine less accurate initial camera poses, as studied in DeepSfM [16]. On domain-specific datasets, fine-tuning can yield an improvement, e.g., 1.7% better on the metric δ_1 on KITTI Depth.

2.3. Other parameters

Essential Matrix. In our framework, we compute initial relative camera poses from essential matrices computed from dense optical flow matches using the 5-point algorithm embedded in a RANSAC framework for robustness. Similarly to [14], we set the inlier error threshold $\tau = 0.0001$ and the number of iterations $\theta = 5$ for KITTI Depth and $\theta = 20$ for static datasets MVS, Scenes11, SUN3D and multi-body datasets ETH3D and Multi-body Unstructured.

Depth and Pose Estimation Network. For KITTI Depth, we adopt 128 3D virtual planes in our depth estimation

Figure 3: **Multi-body Unstructured Extract.** Three view pairs extracted from our proposed dataset for illustration purposes. The dataset contains indoor sequences that depict varied scenarios in which objects of several different sizes move in the scene.

module for each detected motion. For all other datasets, this is set to 96. We set $d_{\text{min}} = 0.5$ the distance between successive virtual depth planes in MVS, Scenes11, SUN3D, ETH3D and Multi-body Unstructured. Instead, for KITTI Depth we set $d_{\text{min}} = 1.0$.

Loss Function Weights. For training, the loss function is: $\mathcal{L} = \lambda_r \mathcal{L}_{\text{rot}} + \lambda_t \mathcal{L}_{\text{trans}} + \lambda_d \mathcal{L}_{\text{depth}}$, with weights $\lambda_r = 0.8$, $\lambda_t = 0.1$ and $\lambda_d = 0.1$ balanced for the optimal trade-off between depth and camera pose according to [16].

2.4. Network Selection

In our framework, we use the optical flow network DICL-Flow [15] for initial up-to-scale camera pose estimation. Our novel depth estimation network builds upon the original architecture of DPSNet [9] with the addition of our original *multi-body* plane sweep algorithm for depth cost volume construction. Finally, we use the monocular depth estimation network AdaBins [5] with pre-trained weights on the KITTI and the NYUv2 datasets provided by the authors.

2.5. Timings

As in [14], the depth module is the main bottleneck, thus adding a traditional motion segmentation step affects inference times only marginally. In KITTI, [14] takes 0.68 ms and our method 0.79 ms: 067 0.05 (flow) + 0.21 (Pose + Motion Segmentation) + 0.53 (Depth).

3. The Multi-body Unstructured dataset

In this section, we describe the proposed *Multi-body Unstructured* dataset introduced in Sec. 5.1 for depth and camera pose evaluation in the multi-body setting. The dataset comprises 21 multi-body scenes of indoor environments for a total of 42 image frames, six of which are illustrated in Fig. 3 for reference. Frames contain either 2 or 3 moving objects. The images are acquired using a Kinect v1 camera with a resolution of 640×480. Depth maps are also 640×480 and are expressed in millimeters from the camera viewpoint. Although not critical to multi-view stereo performance, we calibrate our sensor with the MATLAB R2021b Camera Calibration app included in the Computer Vision toolbox using 20 shots of a 10×7 checkerboard pattern. We obtain the following calibration matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{pmatrix} 589.18 & 0 & 321.15 \\ 0 & 589.75 & 235.55 \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

Ground truth camera poses are obtained using the state-of-the-art RGB-D SLAM software [7] from video sequences of approximately 10 seconds. As discussed in the main manuscript, this software fuses visual information and data provided by the Kinect IMU sensor to provide very accurate camera poses. We extract two frames from each video sequence by considering pairs of frames with fairly large camera baselines and significant object movements.

In this Supplementary Material we include sequences used for the evaluation of ours and competing methods. The dataset is organized similarly to the *static* sequences from DeMoN [13] and uses the same format introduced in DPSNet and DeepSfM [9, 16] to make testing easier. We describe the format of our dataset as follows. Each sequence in the dataset is numbered, e.g. $00001, 00002, \ldots$, and contains two RGB images, 0000.jpg and 0001.jpg, and their respective ground truth depths, 0000.npy and 0001.npy.

Fig. 4 shows a larger version of Fig. 5 in the main paper, which includes images, ground truth depth maps and depth estimation results from our method and DeepSfM [16]. Fig. 4 also includes results from Wang et al. [14], which were omitted from the main paper due to space restrictions.

The *dataset* folder in the .zip file contains the sequences used for testing in the aforementioned format.

4. Depth Metrics

In this section, we define the metrics used in the main manuscript for depth evaluation. For KITTI Depth, we adopt the metrics in [6]. Given an estimated depth map $\mathbf{D}$ and its corresponding ground truth $\mathbf{D}_{\text{gt}}$, y is a pixel belonging to $\mathbf{D}$ and y^* is a pixel at the same coordinates of y belonging to $\mathbf{D}_{\text{gt}}$. Specifically, we consider the depth Absolute Relative difference (Abs Rel):

$$\text{Abs Rel} = \frac{1}{|\mathbf{D}|} \sum_{y \in \mathbf{D}} |y - y^*| / y^*, \tag{2}$$

	MVS					Scenes11					SUN3D				
	Depth			Pose		Depth			Pose		Depth			Pose	
Method	L1-inv	Sc-inv	L1-rel	$\mathbf{R}_{\text{err}}$	t_{err}	L1-inv	Sc-inv	L1-rel	$\mathbf{R}_{\text{err}}$	t_{err}	L1-inv	Sc-inv	L1-rel	$\mathbf{R}_{\text{err}}$	t_{err}
Base-SIFT	0.056	0.309	0.361	21.180	60.517	0.051	0.900	1.027	6.179	56.650	0.029	0.290	0.286	7.702	41.825
Base-Matlab	-	-	-	10.843	32.736	-	-	-	0.917	14.639	-	-	-	5.920	32.298
COLMAP [11]	-	-	0.384	7.961	23.469	-	-	0.625	4.834	10.682	-	-	0.623	4.235	15.956
Ours ($M = 1$)	0.016	0.107	0.068	2.538	4.538	0.005	0.099	0.058	0.321	3.649	0.011	0.084	0.061	1.470	12.018
Ours ($M = 2$)	0.019	0.121	0.073	2.681	7.340	0.007	0.102	0.069	0.401	4.619	0.013	0.092	0.071	1.625	13.402
Ours ($M = 3$)	0.025	0.132	0.075	2.892	9.530	0.009	0.124	0.078	0.542	5.782	0.014	0.095	0.079	1.769	15.231
Ours ($M = 4$)	0.026	0.134	0.076	2.931	9.741	0.009	0.128	0.081	0.571	5.803	0.016	0.107	0.084	1.830	15.904

Table 2: **Depth and pose evaluation on MVS, Scenes 11, SUN3D.** Base-SIFT and Base-Matlab come from [13]. For all metrics, lower is better. Best results are in **bold**.

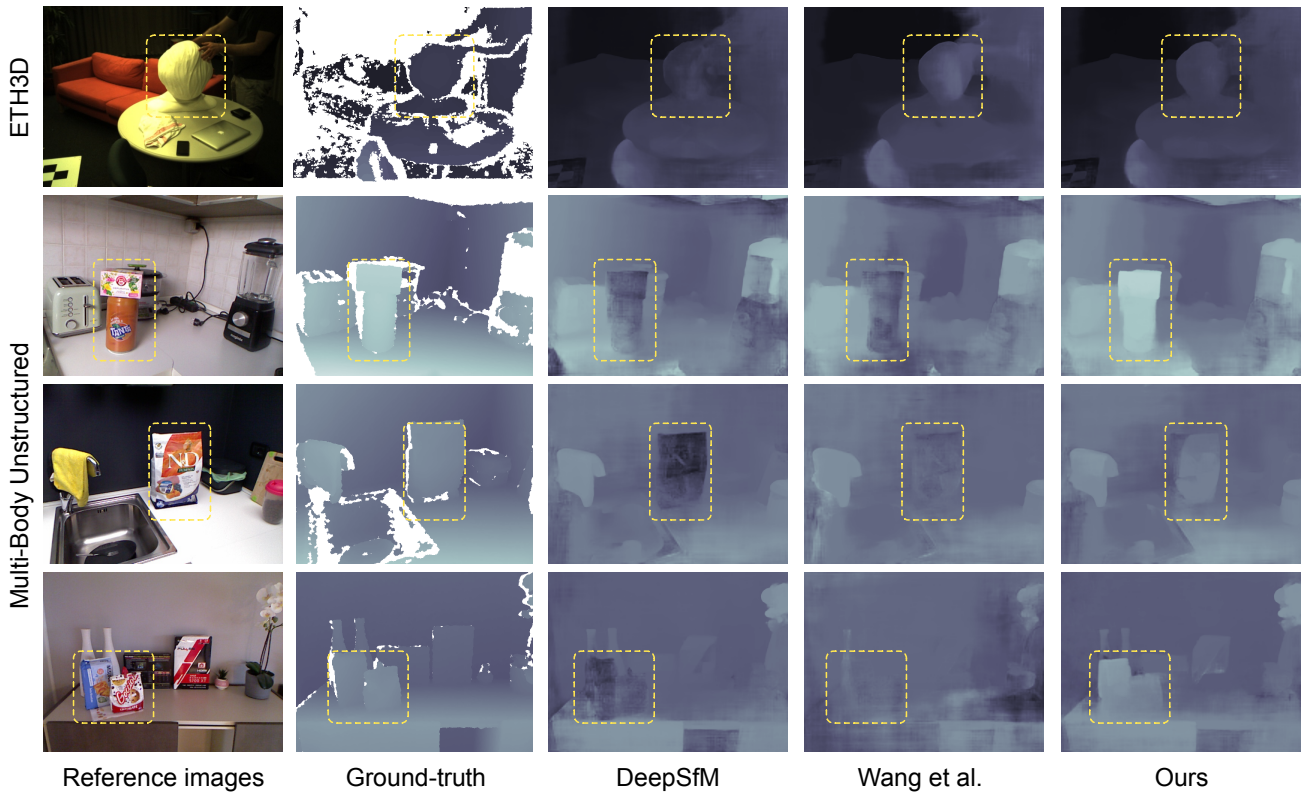

Figure 4: **Qualitative comparisons on ETH3D and Multi-body Unstructured.** We compare our method against DeepSfM [16] and Wang et al. [14], which was omitted from the comparison in the main paper due to space restrictions. The yellow boxes highlight the moving objects in the scene that are successfully reconstructed by our method but not by its competitors.

the depth Squared Relative difference (Sq Rel):

$$\text{Sq Rel} = \frac{1}{|\mathbf{D}|}\sum_{y\in\mathbf{D}} ||y - y^*||^2/y^*, \tag{3}$$

the Root Mean Square Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{|\mathbf{D}|}\sum_{y\in\mathbf{D}} ||y - y^*||^2}, \tag{4}$$

the log Root Mean Square Error ($\text{RMSE}_{\log}$):

$$\text{RMSE}_{\log} = \sqrt{\frac{1}{|\mathbf{D}|}\sum_{y\in\mathbf{D}} ||\log(y) - \log(y^*)||^2}, \tag{5}$$

and, finally, the threshold accuracy δ_i, i.e., the $\%$ of y such that:

$$\max(\frac{y}{y^*}, \frac{y^*}{y}) = \delta < thr, \tag{6}$$

where $thr = 1.25$ for δ_1, $thr = 1.25^2$ for δ_2, and $thr = 1.25^3$ for δ_3.

For all other datasets, we consider the metrics in [13], i.e., the scale-invariant depth error (sc-inv):

$$\text{sc-inv} = \sqrt{\frac{1}{|\mathbf{D}|}\sum_{y\in\mathbf{D}}(y - y^*)^2 - \frac{1}{|\mathbf{D}|^2}(\sum_{y\in\mathbf{D}} y - y^*)^2}, \tag{7}$$

the L1 relative error (L1-rel), which is equivalent in its formulation to the aforementioned Abs Rel, and the inverse L1 error (L1-inv):

$$\text{L1-inv} = \frac{1}{|\mathbf{D}|}\sum_{y\in\mathbf{D}}|\frac{1}{y^*}-\frac{1}{y}|. \tag{8}$$

5. Additional Experiments

Our method is primarily designed to work on scenes with rigidly moving objects and sparse unstructured images. Nonetheless, we found it interesting to investigate the behavior of our framework when operating under sub-optimal conditions based on these assumptions. After having demonstrated that our approach outperforms traditional methods on static scenes, in Sec. 5.2, we examine what happens in the presence of articulated motions, and in Sec. 5.3, we compare our approach to dense methods that either benefit from small baselines between image pairs or exploit video temporal information.

5.1. Static Evaluation against Traditional Baselines

As mentioned in Sec. 5.2 of the main manuscript, our multi-body framework outperforms traditional baselines even in static scenes. In this regard, Tab. 2 compares our method against static SfM pipelines on the single-body MVS, Scenes11 and SUN3D datasets. Specifically, we consider the Base-SIFT and Base-MATLAB reported in [13], which estimate depth and camera poses by SIFT features and KLT tracking correspondences respectively, and COLMAP [11], a well known SfM pipeline. It can be appreciated that our methods produces significantly more accurate depth and camera pose in all metrics for all the datasets.

5.2. Dealing with Articulated Motions

We performed some qualitative experiments on the articulated *"arm"* scene from the Hopkins-155 dataset [12] which consists of $\mu = 3$ dominant motions in the scene, where β_1 is the static background and chest of the person, β_2 is the moving arm, and β_3 is the chessboard object that moves independently with respect to the arm holding it. The results, in Fig. 5, are promising and, as stated in Sec. 5.4 of the main manuscript, our method can reconstruct articulated motions provided that an adequate number of motions M is considered.

We compare the results of our method against the state-of-the-art DeepSfM [16]. DeepSfM, which works under the assumption that the scene is static, reconstructs only the static background and person β_1 and produces depth maps with artifacts in the image region covered by the arm β_2 and the chessboard object β_3. However, our method can reconstruct β_2, highlighted in the red box, in the $M = 2$ configuration and both β_2 and β_3, highlighted in the yellow box, in the $M = 3$ configuration. Our method can thus reconstruct articulated motions, provided that a sufficient number of motions is considered.

As discussed in Sec. 5.4 of the main manuscript, the upper limit on the number of rigid motions M affects the number of depth planes allocated to each of the $\{\beta_k\}_{k=1}^{M}$ objects. When considering scenes with many moving bodies or articulated motions, it is thus necessary to increase the number of virtual depth planes L to preserve accuracy in depth estimation, with the main drawback of slightly increased inference times. Specifically, on our test system consisting of 2 NVIDIA A6000 GPUs, inference takes approximately 2.31 seconds with $M = 2$ and 2.52 seconds with $M = 3$.

5.3. Dealing with Dense Image Sequences

Our method is designed to cope primarily with unstructured images without assuming temporal coherence. Nonetheless, it is interesting to compare against methods that work on video streams or with small baselines for optimal performance. To this end, we evaluate our method on the KITTI VO dataset [8], which contains 10 video sequences with ground truth camera poses. As in [14], in our experiments we consider the frames from the left camera in sequences "09" and "10" for a fair comparison. Most frames in these sequences depict a scene without moving objects, thus we also consider a subset of these sequences to evaluate our method and other state-of-the-art camera pose estimation approaches in the multi-body setting. Specifically, we consider two splits: *i)* the full-length videos, namely Seq. 09 *(all)* and Seq. 10 *(all)*, *ii)* a subset of the video frames in which dynamic objects appear, namely Seq. 09 *(MB)* and Seq. 10 *(MB)*. The total number of frames considered for multi-body camera pose evaluation is 102.

As in [17], we measure the pose estimation accuracy on relative translational error t_{err}, expressed in percentage, and relative rotational error r_{err}, in $\deg/100m$. The predicted trajectories are aligned to the ground truth via least square optimization as in [14].

Tab. 3 shows the results attained by our method, SfMLearner [17], CCNet [10] and Wang et al. [14], which have been introduced in the main manuscript, on the aforementioned sequences. Our method outperforms SfMLearner and CCNet by a significant margin on all the considered sequences. Compared to Wang et al. [14], our method attains worse performance by 6.9% on average on the entire Seq. 09 *(all)* and Seq. 10 *(all)*, but performs better in the multi-body sub-sequences Seq. 09 *(MB)* and Seq. 10 *(MB)* by 4.2% on average.

As expected, as the majority of frames in Seq. 09 *(all)* and Seq. 10 *(all)* depict a scene with a single motion, some multi-view methods operating under the static scene assumption achieve more accurate results. As discussed in the main manuscript, our method is configured to handle

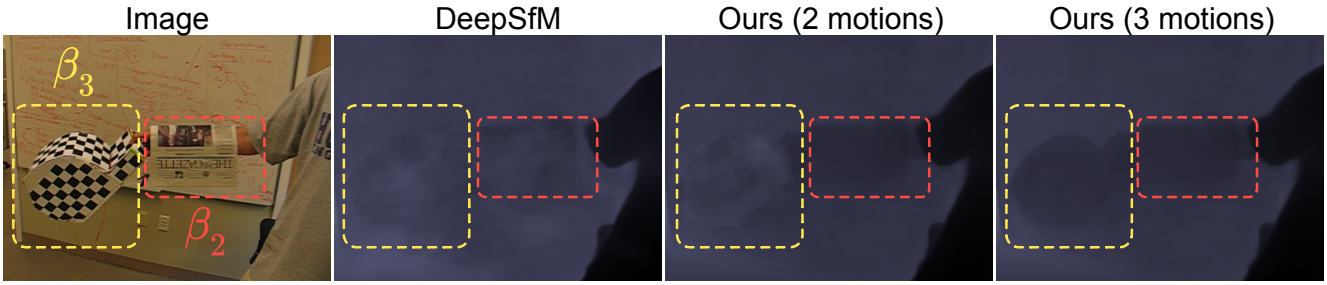

Figure 5: **Qualitative evaluation on Hopkins-155.** We compare our method against DeepSfM [16] on the *"arm"* sequence from Hopkins-155. Our method is configured to handle either $M = 2$ or $M = 3$ rigid motions in the scene. DeepSfM cannot reconstruct the rigidly moving objects β_2 (red box) and β_3 (yellow box). Instead, our method can reconstruct β_2 when $M = 2$ and both β_2 and β_3 when $M = 3$.

	Seq. 09 *(all)*		Seq. 09 *(MB)*		Seq. 10 *(all)*		Seq. 10 *(MB)*	
Method	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}
SfMLearner [17]	8.28	3.07	8.97	3.51	12.20	2.96	13.04	3.84
CCNet [10]	6.92	1.77	-	-	7.97	3.11	-	-
Wang et al. [14]	**1.70**	**0.48**	2.04	0.61	**1.49**	**0.55**	1.76	0.68
Ours	1.81	0.52	**1.98**	**0.59**	1.57	0.59	**1.69**	**0.64**

Table 3: **Pose evaluation on KITTI VO.** We consider the full sequences "09" and "10" (Seq. 09 *(all)* and Seq. 10 *(all)*) as well as a subset of the frames of the aforementioned sequences in which dynamic objects appear (Seq. 09 *(MB)* and Seq. 10 *(MB)*). For all metrics, lower is better. Best results in **bold**.

up to $\mu = 3$ motions, which results in outlying essential matrices that may impact camera pose estimation. Instead, in the challenging multi-body setting, our method achieves significantly more accurate results than its competitors in camera pose estimation. This validates our claim that, by segmenting the motions in the scene before estimating essential matrices, our method is extremely robust to moving objects when regressing relative camera poses.

6. Code

We publicly provide the implementation in Python and PyTorch of our multi-body plane-sweep depth estimation network at https://github.com/andreadalcin/MultiBodySfM. In our implementation, we use the following open source code: *i*) AdaBins for monocular depth estimation [5], *ii*) DICL-Flow [2] for optical flow, *iii*) multi-frame motion segmentation [1], *iv*) DPSNet [3] depth estimation network, *v*) DeepSfM [4] pose estimation network.

References

[1] https://github.com/federica-arrigoni/ICCV_19. 6

[2] https://github.com/jytime/DICL-Flow. 6

[3] https://github.com/sunghoonim/DPSNet. 6

[4] https://github.com/weixk2015/DeepSFM. 6

[5] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4009–4018, 2021. 3, 6

[6] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014. 3

[7] Felix Endres, Jürgen Hess, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. 3-d mapping with an rgb-d camera. *IEEE transactions on robotics*, 30(1):177–187, 2013. 3

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012. 5

[9] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. Dpsnet: End-to-end deep plane sweep stereo. *arXiv preprint arXiv:1905.00538*, 2019. 2, 3

[10] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12240–12249, 2019. 5, 6

[11] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4, 5

[12] Roberto Tron and René Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007. 5

[13] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5038–5047, 2017. 3, 4, 5

[14] Jianyuan Wang, Yiran Zhong, Yuchao Dai, Stan Birchfield, Kaihao Zhang, Nikolai Smolyanskiy, and Hongdong Li. Deep two-view structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8953–8962, 2021. 2, 3, 4, 5, 6

[15] Jianyuan Wang, Yiran Zhong, Yuchao Dai, Kaihao Zhang, Pan Ji, and Hongdong Li. Displacement-invariant matching cost learning for accurate optical flow estimation. *Advances*

in Neural Information Processing Systems, 33:15220–15231, 2020. 2, 3

[16] Xingkui Wei, Yinda Zhang, Zhuwen Li, Yanwei Fu, and Xiangyang Xue. Deepsfm: Structure from motion via deep bundle adjustment. In *European conference on computer vision*, pages 230–247. Springer, 2020. 2, 3, 4, 5, 6

[17] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858, 2017. 5, 6