Supplementary Material for
The Drunkard's Odometry:
Estimating Camera Motion in Deforming Scenes

David Recasens
University of Zaragoza

Martin R. Oswald
ETH Zurich, University of Amsterdam

Marc Pollefeys
ETH Zurich, Microsoft

Javier Civera
University of Zaragoza

Abstract

This supplementary document provides additional details about the dataset, the method, the evaluation and further experimental results.

Contents

A Drunkard's Dataset Details ……………………………………………………………… 1
B Drunkard's Odometry Flow……………………………………………………………… 2
C Drunkard's Odometry Ablations………………………………………………………… 3
D Inference speed comparison of the baselines ………………………………………… 4
E Architecture Details …………………………………………………………………… 4

A Drunkard's Dataset Details

Table A.1 details the number of frames for each individual scene of the Drunkard's Dataset[1], and the total one. The number of frames is equal across all difficulty levels (hence the $4\times$) in the table. Aggregating all the levels, the Drunkard's Dataset has more than 400K high-resolution frames. In order to enhance its suitability with different computing setups and to reduce its overall size, the dataset is made available in two different image resolutions, 1024×1024 and 320×320 pixels. Other resolutions can be easily regenerated using the provided scripts. The Drunkard's Dataset is built by animating the 3D meshes of the Habitat-Matterport 3D dataset [1] (that is MIT licensed) and we use the same scene numbering. Scene 17 was skipped since it did not contain an appropriate 3D model to create an exploratory camera trajectory in it. The Blender files are also made public with scripts to replicate the rendering, so additionally anyone can create their custom datasets with different deformations, resolution, camera parameters or trajectory for example.

Regarding the deformations applied to the 3D meshes in Blender to obtain the dynamic non-rigidity of the scene and the perturbations to the camera trajectory, the Table A.2 collects those whose values vary between the four different levels of difficulty. The three mesh deformations were all applied to three different empty planes that are flying randomly throughout the 3D space around the mesh. In consequence, at a certain timestamp, the higher the proximity of a plane to a surface, the higher the amplitude of the rendered deformation. The camera perturbations are Gaussian noise that adds to the rotation and translation of the manually recorded steady camera trajectory. All these mesh and

---

[1]https://davidrecasens.github.io/TheDrunkard'sOdometry/

37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks.

1

Scene	# Frames	Scene	# Frames	Scene	# Frames
0	4×3.816	7	4×2.341	14	4×23.863
1	4×1.523	8	4×3.033	15	$4 \times\ 7.984$
2	4×1.240	9	4×8.098	16	$4 \times\ 5.389$
3	4×3.308	10	4×2.632	18	4×10.989
4	4×4.545	11	4×4.168	19	$4 \times\ 9.739$
5	4×1.655	12	4×4.168	**Total**	$\mathbf{4 \times 104.302}$
6	4×1.515	13	4×3.296		

Table A.1: **Drunkard's dataset overview.** The datasets consists of more than 400K frames distributed over 20 scenes and 4 difficulty levels. The table lists the number of frames for every scene and the total for one of the four ($4\times$) difficulty levels.

camera parameters increase in amplitude as difficulty grows. All the Blender project files for every scene and level of difficulty are publicly available on the website, so anyone can render customized versions of the Drunkard's Dataset and see the full details of the implementation.

		Difficulty level			
	Parameter	0	1	2	3
Mesh	Cast factor	0	0.01	0.03	0.05
	Wave height	0	0.05	0.1	0.15
	Simple deform angle [°]	0	0.5	1	1.5
Camera	Noise strength rotation	0	0.6	1.2	2
	Noise strength traslation	0	0.3	0.6	1

Table A.2: **Blender's 3D mesh deformation and camera trajectory perturbation parameters** along the four levels of difficulty of the Drunkard's Dataset.

B   Drunkard's Odometry Flow

Table B.1 shows 2D and 3D flow evaluation metrics for Drunkard's Odometry and its inspirational work RAFT-3D [2] (which is BSD 3-clause licensed) in all the four levels of the same four test scenes of the Drunkard's Dataset. As already said in the main paper, RAFT-3D exclusively predicts 3D scene flow, while we additionally estimate the camera pose. We evaluate both methods in terms of flow metrics in order to validate our implementation.

Both methods are trained from scratch during 10 epochs on the same Drunkard's Dataset scenes (all except the test ones) and with the same deformation (difficulty level 1). The base evaluations metrics are the 2D and 3D End-Point-Error (EPE). EPE is computed as the mean Euclidean distance between the estimated and ground truth flow (**flow** and $\overline{\textbf{flow}}$, respectively) in all valid pixels of the N test images

$$\text{EPE} = \frac{\sum_{i=1}^{N}\|\textbf{flow} - \overline{\textbf{flow}}\|_2}{N} \ , \tag{1}$$

In the 2D case, **flow** is the optical flow that we denote in the main paper as $\textbf{O}$. In the 3D case, **flow** is the 3D scene flow $\textbf{flow}_{3D}$ that comes from back-projecting every pixel $j \in \textbf{I}_1$ with coordinates $\textbf{u}_j$ and its corresponding in camera 2 $\textbf{u}_{j\prime}$, associated by the optical flow $\textbf{O}$, to the 3D space using their sensor depth values

$$\textbf{flow}_{3D} = \textbf{P}_{j\prime} - \textbf{P}_j = \pi^{-1}\left(\textbf{u}_{j\prime}, \bar{z}_{j\prime}\right) - \pi^{-1}\left(\textbf{u}_j, \bar{z}_j\right) \ . \tag{2}$$

We exclude invalid pixels which have excessively large optical flow module values (> 250), infinite depths (as they define holes in the 3D mesh), as well as pixels for which correspondences fall outside the image boundaries.

2

The δ metrics are the percentage of the total pixels in the test scenes that have an EPE value under a certain threshold, being 1 pixel for the 2D case and 1 cm for the 3D case. In the 3D metrics, $|\Delta\text{EPE}|$ [%] stands for the absolute difference in percentage between the 3D EPE of RAFT-3D and of Drunkard's Odometry.

Scene	Level	Method	2D Metrics		3D Metrics		
			$\delta < 1$ px [%]$\downarrow$	EPE [px]$\downarrow$	$\delta < 1$ cm [%]$\downarrow$	EPE [mm]$\downarrow$	$|\Delta\text{EPE}|$ [%]
0	0	RAFT-3D	93.1	0.43	96.1	2.77	32.7
		Drunkard's Odometry	91.5	0.56	95.0	3.67	
	1	RAFT-3D	91.8	0.48	95.4	3.06	33.8
		Drunkard's Odometry	89.5	0.63	94.1	4.11	
	2	RAFT-3D	87.2	0.69	93.1	3.92	34.3
		Drunkard's Odometry	83.7	0.88	90.9	5.27	
	3	RAFT-3D	79.1	1.29	89.2	5.52	32.5
		Drunkard's Odometry	74.2	1.54	85.7	7.32	
4	0	RAFT-3D	83.3	1.62	86.8	9.58	28.0
		Drunkard's Odometry	79.3	2.09	83.6	12.27	
	1	RAFT-3D	83.4	1.36	86.8	9.20	31.6
		Drunkard's Odometry	79.2	1.80	83.2	12.11	
	2	RAFT-3D	81.6	1.42	84.8	9.76	31.5
		Drunkard's Odometry	76.6	1.87	80.3	12.82	
	3	RAFT-3D	77.4	1.77	81.6	11.13	32.43
		Drunkard's Odometry	71.0	2.30	75.7	14.74	
5	0	RAFT-3D	85.7	1.16	89.1	8.82	27.7
		Drunkard's Odometry	82.3	1.53	86.1	11.27	
	1	RAFT-3D	85.4	1.15	87.6	9.08	25.1
		Drunkard's Odometry	81.7	1.49	84.9	11.37	
	2	RAFT-3D	82.1	1.29	84.9	10.26	26.1
		Drunkard's Odometry	77.1	1.65	80.3	12.94	
	3	RAFT-3D	75.7	1.76	79.9	12.09	30.21
		Drunkard's Odometry	69.19	2.19	74.2	15.74	

Table B.1: **2D and 3D flow metrics** in each of the four levels of difficulty of the three test scenes of the Drunkard's Dataset for RAFT-3D and our Drunkard's Odometry.

From EPE and δ metrics in Table B.1 we conclude that our Drunkard's Odometry, even focused on camera pose and not on flow as RAFT-3D, performs close to RAFT-3D on these metrics. As RAFT-3D estimates only flow, it is not as affected as our Drunkard's Odometry by the different levels of deformation. The stability of the $|\Delta\text{EPE}|$ [%] values for the four difficulty levels at each scene shows that the 3D flow estimation of our Drunkard's Odometry is stable. This gives evidence about the capability of our method to simultaneously estimate accurate scene flow $\mathbf{T}$ and camera poses $\mathbf{T}_c$ independently of the deformations, as the total flow $\mathbf{T}$ includes the flow caused by scene deformations $\mathbf{T}_d$ and the one caused by camera motion $\mathbf{T}_c$.

C   Drunkard's Odometry Ablations

We tried several configurations of our Drunkard's Odometry to find the best training setup, which was possible thanks to our generated dataset. Figure C.1 shows the Relative Position Error (RPE) metric for translation and rotation (without trajectory alignment as the training dataloader samples random adjacent pair of frames) in the three Drunkard's Dataset test scenes at level of difficulty 1 for different variations of the Drunkard's Odometry. All the methods were trained during 10 epochs from scratch, but varying training parameters. Specifically, we played with the learning rate (l.r., 10^{-5}, 10^{-4} and 2×10^{-4}), inverting the order of the input image pair randomly (with 50% probability), and different loss hyperparameters. We also trained models deactivating the intermediate optical flow $\mathbf{O}^{k,\text{pre}}$ loss term with w_1, changing the influence of the pre-estimate of the camera pose $\mathbf{T}_{\text{pose}}^{\text{pre}}$ w_4, cancelling all

3

supervision from optical flow and depth ground truth leaving only the pose loss, and modifying the relative weight w_{rot} between translation and rotation in the pose loss (that by default is 1).

$$\|\log_{\text{SE(3)}}\left(\mathbf{T}_c^{\text{pre}}\bar{\mathbf{T}}_c^{-1}\right)\|_1 = \|\log_{\text{SE(3)}}\left(\mathbf{T}_c^{\text{pre}}\bar{\mathbf{T}}_c^{-1}\right)\|_1^{\text{tra}} + w_{\text{rot}}\|\log_{\text{SE(3)}}\left(\mathbf{T}_c^{\text{pre}}\bar{\mathbf{T}}_c^{-1}\right)\|_1^{\text{rot}} \ . \quad (3)$$

We observed that our model can be trained at a faster learning rate (2×10^{-4}) than RAFT-3D (10^{-4}), possible due to the additional supervision. Our models need optical flow and depth supervision, in addition to pose, for convergence (without all these, the training diverges eventually). Inverting the order of the input image pair during training, as a data augmentation technique, has a significant impact in reducing the translation pose estimation error (notice how the translation curve keeps pointing down even in the last training stages, which does not happen without such augmentation).
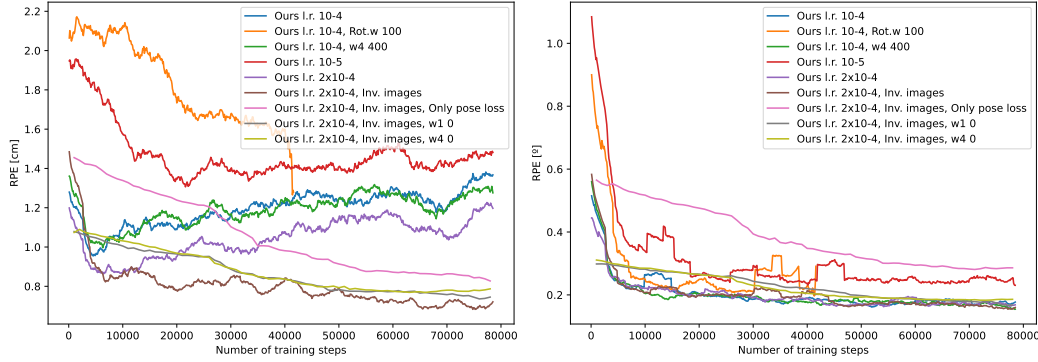
Figure C.1: **Ablations and hyperparameter influence on model training.** We show the relative position errors (RPE) in the three Drunkard's Dataset test scenes for difficulty level 1 and for different model variations of our Drunkard's Odometry during training. With "l.r." we denote different learning rates. If applied, data augmentation with inverted images is shown as "inv. images" and different loss term weights are shown as "w1", "w4", "Rot. w" followed by their values.

D   Inference speed comparison of the baselines

In Table D.1 are the approximate inference times per frame for all the evaluated baselines previously. These values come from rounding the total time taken for estimating the full trajectory of Scene 5 of the Drunkard's Dataset with resolution of 320×320 pixels at level of difficulty 0 divided by the total number of frames (1.655). Our Drunkard's Odometry is the most efficient of all, in addition to being the most accurate. Note that, even if our Drunkard's Odometry is based on iterative refined optical flow like DROID-SLAM, in our case we do not perform global Bundle Adjustment, which is what gives us the computational advantage. Further note that Scene 5 is the shortest one among the three tested (0, 4 and 5). For longer sequences, the gap between Drunkard's Odometry versus DROID-SLAM and Colmap would be higher.

Method	Time/frame [ms]
Colmap	1.650
DROID-SLAM	200
EDaM	700
Drunkard's Odometry	**170**

Table D.1: **Inference time per frame** for all the baselines.

E   Architecture Details

The architectural details for the feature encoder, the context encoder, the pose network and the update block of our network are shown in Figures E.1 and E.2. The figures show network blocks (in gray)

4

and input, intermediate and output variables (in blue, green and red respectively). The pose network is a modification of the one used in ManyDepth [3] (Copyright © Niantic, Inc. 2021. See project website for more license details), so the initial guess of the relative camera pose given by the pose network is in Lie algebra ($\mathbf{t}_c^{\text{pre}}$) and it is exponential mapped back to SE(3) ($\mathbf{T}_c^{\text{pre}}$) to initialize $\mathbf{T}_c$ and $\mathbf{T}$ before the iterative block. In the update block, the input pose twist $\mathbf{t}_c^k$ is cloned to every pixel value of a tensor with the same resolution as the other input variables ($H/8 \times W/8$) to be able to stack them together.

(a) Feature encoder   (b) Context encoder   (c) Pose network
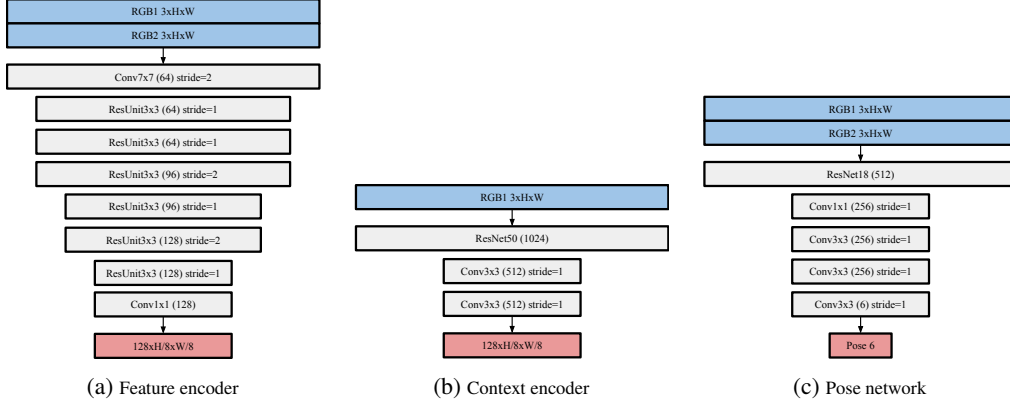
Figure E.1: **Network details** for the feature encoder, the context encoder and the pose network.
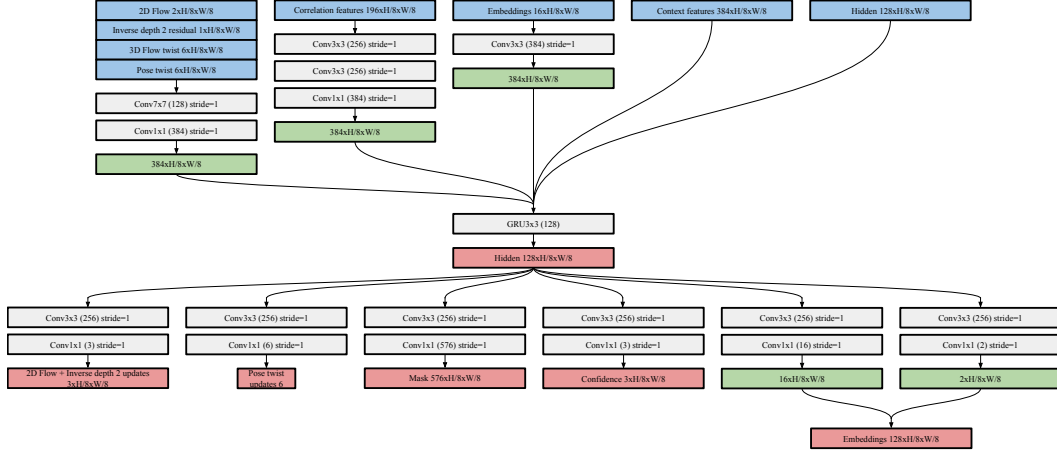
Figure E.2: **Network details for the update block.**

References

[1] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang, *et al.*, "Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai," *arXiv preprint arXiv:2109.08238*, 2021. 1

[2] Z. Teed and J. Deng, "Raft-3d: Scene flow using rigid-motion embeddings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8375–8384, 2021. 2

[3] J. Watson, O. Mac Aodha, V. Prisacariu, G. Brostow, and M. Firman, "The temporal opportunist: Self-supervised multi-frame monocular depth," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1164–1174, 2021. 5

5