Deep Patch Visual Odometry Supplementary Material

Anonymous Author(s)
Affiliation
Address
email

A Stable Runtime

The runtime of our approach is relatively constant compared to prior VO/SLAM systems. Our default configuration averages 60FPS, and rarely drops below 50FPS. In Fig. A, we show the runtime distribution on EuRoC. The runtime stability of DPVO results from the comparatively simple keyframing mechanism.

Discussion on Keyframing: Most VO/SLAM systems (1; 6; 11), including DROID-SLAM, contain a protocol for deciding when to produce new keyframes from a video stream. Keyframing mechanisms serve to discard repetitious frames early-on, saving on computation and potentially improving performance by limiting redundancy. Many VO/SLAM systems require a low keyframing frequency in order to run at real-time framerates. If there is significant motion between subsequent frames, keyframes are created at a high frequency which can cause such methods to slow down drastically. DROID-SLAM, for instance, slows from 40FPS to 11FPS due to an increase in the number of keyframes being created.

Unlike previous works, DPVO treats *all* incoming frames as keyframes and only removes redundant frames later using motion computed from the already-estimated pose. Although this design decision is sub-optimal (in terms of speed) during very slow or still camera movement (e.g. EuRoC), our approach is both simpler and ensures our frame-rate is approximately constant no matter the degree of camera motion.

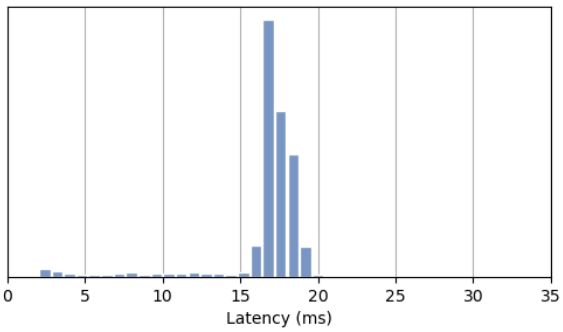
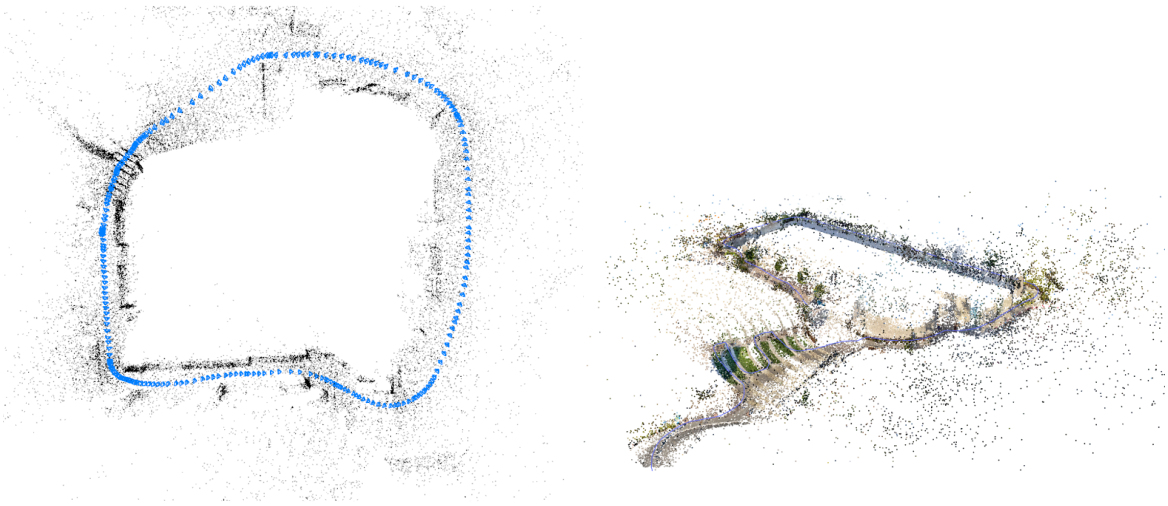


Figure A: Runtime distribution on EuRoC. Our system averages 60FPS with each new frame taking ∼17ms to process. The distribution is very centralized and rarely drops below 50FPS.

B ATE Error Metric

We compare results using the ATE (average trajectory error) metric, which is standard for VO and SLAM (2; 11; 6; 13; 1). The ATE metric between the predicted trajectory $\tilde{\mathbf{T}}$ and the ground-truth

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.



(a) Reconstruction on the ETH-3D (9) dataset. (b) Reconstruction from our video demo.

Figure B: Sparse reconstructions produced by our model.

trajectory $\hat{\mathbf{T}}$ is computed using

$$E(\tilde{\mathbf{T}}, \hat{\mathbf{T}}) := \sum_{t=1}^{N} ||\tilde{\mathbf{T}}_{xyz} - \hat{\mathbf{T}}_{xyz}||_2 \tag{1}$$

after aligning the two trajectories using a similarity transformation, to account for the scale and SE(3) gauge freedoms. This metric is computed using the EVO library (3).

C Additional Training Details

DPVO is trained entirely on TartanAir (14)[1], a synthetic dataset. This is the same synthetic training data used by previous VO systems (13; 11). TartanAir provides a large number of training scenes of both indoor and outdoor environments with varied lighting and weather conditions. The dataset provides depth and camera pose annotations, enabling one to generate optical flow by re-projecting the depth using the camera poses. All dataset parameters are identical to those used in DROID-SLAM (11). This includes random cropping, color-jitter, random greyscale, random color inversion, as well as re-scaling the training scenes in order to improve training stability.

D Visualization

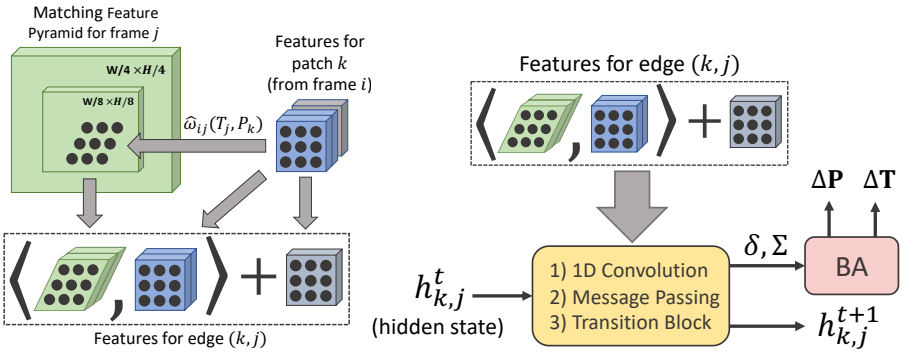
Reconstructions are visualized interactively using a separate visualization thread. Our visualizer is implemented using the Pangolin library[2]. It directly reads from PyTorch tensors avoiding all unnecessary memory copies from CPU to GPU. This means that the visualizer has very little overhead– only slowing the full system down by approximately 10%. In Fig. Ba, we show a reconstruction on the ETH-3D (9) dataset, and in Fig. Bb we show the reconstruction produced during our video demo.

E Bundle Adjustment Layer

Our bundle adjustment layer is functionally identical to the BA layer in DROID-SLAM (11), except that DROID-SLAM predicts the damping factor using its update operator, while we found this not to be necessary and hard-code it to 10^{-4} instead. We refer the reader to Appendix C in the

[1] `https://theairlab.org/tartanair-dataset/`

[2] `https://github.com/stevenlovegrove/Pangolin`



(a) Correlation Step (b) Update Operator

Figure C: The update operator, including the correlation step. **(a)** For each edge (k, j) in the patch graph, patch k is reprojected into a frame j using equation (2). Frame j's matching features are then cropped using the reprojected patch. The *correlated* matching features (blue, green) and context features (grey) form the edge features. Each dot in the sampled matching features (green) represents a 7×7 neighborhood of points, however this is omitted for clarity. **(b)** Given the features for edge (k, j), several learnable layers (yellow) are used to predict an update $\delta_{(k,j)} \in \mathbb{R}^2$ to the flow, a confidence weight $\Sigma_{(k,j)} \in \mathbb{R}^2$, and an update to the hidden state $h^t_{k,j}$. The predicted factors are used in the bundle adjustment layer to produce an update to the camera poses and patch depths. The factor head is omitted here for clarity.

DROID-SLAM paper for the analytical gradients of the Gauss-Newton update iterations. Since DROID-SLAM uses dense flow, we implement our own sparse version as an optimized CUDA layer.

F Additional Figures

Correlation Operation: In Fig. Ca, we show a high-level visualization of the correlation operation which is performed at the start of the update operator. Patch k's P^2 matching-features are each projected into the two-level feature pyramid of frame j. A 7×7 grid of $D = 128$ dimension feature vectors are bilinearly sampled from the pyramid around each point reprojection and dot producted with the matching features. This operation results in 2 sets of $p \times p \times 7 \times 7$ dot products, one for each resolution (1/4 and 1/8th of the original image size). This operation is implemented as an optimized CUDA layer.

Update Operator: Following Fig. Ca, in Fig. Cb we show a high-level view of the rest of the update operator. The correlation and context features are fed into additional learn-able layers, including 1D Convolution Layers, a Softmax-Aggregation block (i.e. Message Passing), and a Transition Block. This results in an updated hidden state for each edge. Finally, the updated hidden state is used to produce 2D flow revisions and confidence weights which guide the solution to the bundle adjustment. The bundle-adjustment layer ultimately produces an update to the depths and camera poses.

VO System: In Fig. D we show an overview of the VO system. The visualization (Sec. D) and frame loading are performed in separate threads. The main components of the VO system are discussed in Sec. 3.3 in the main paper.

Confidence Weights: Our approach learns to reject outliers by predicting a confidence weight associated to each predicted 2D flow update, which DROID-SLAM (11) does as well. We show a visualization of the weights for a small number of edges in Fig. E. These weights are not supervised directly, rather they are learned by supervising on the pose output of the differentiable bundle-adjustment layer.

G Future Work

Patch Selection: Prior methods for camera pose estimation may select keypoint locations using heuristics such as image regions with high photometric gradient (1), use deep networks to identify salient keypoints (8), or simply track all $W \times H$ pixels (11; 5; 4).

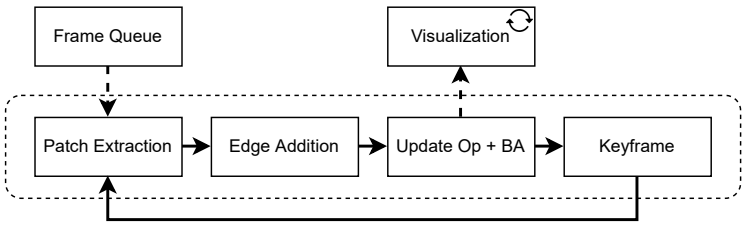


Figure D: Overview of the VO System. The visualization (Sec. D) and frame loading are performed in separate threads. The main components of the VO system are discussed in Sec. 3.3 in the main paper.

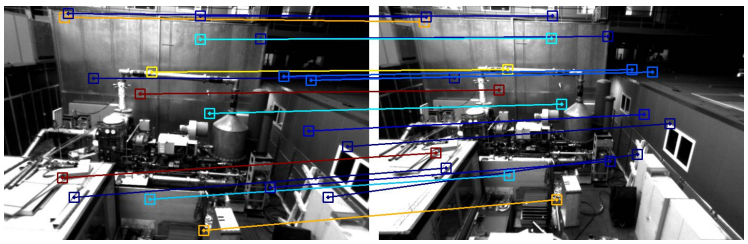


Figure E: Factor confidence weights. For each edge in the patch graph, the factor-head of our update operator predicts a confidence weight $w \in \mathbb{R}^2$ bounded to $(0, 1)$ and a 2D flow update. Cold colors (blue) represent high confidence edges while hot colors (red) represent low confidence edges.

As discussed in our paper ablations, we observe that selecting patch centroids *randomly* outperforms these alternative approaches. While somewhat surprising, we believe this makes sense given the nature of video input. Video frames have considerable overlap, implying that most pixels have a valid match in the adjacent frames. For this reason, the benefits of selecting salient keypoints is potentially outweighted by having a uniform coverage of the whole image. However, there exist a large space of other keypoint selection schemes to explore, which is outside the scope of this work.

Global Optimization: Our paper focuses exclusively on the front-end of the visual SLAM problem. Therefore, an obvious extension of our work would include a back-end with components such as loop closure and global bundle adjustment, composing a full SLAM system. While useful, we consider implementing a back-end orthogonal to our contribution, and we leave this to future work / implementations.

H Limitations

On some sequences, classical methods such as ORB-SLAM (6) and DSO (1) still occasionally out-perform approaches based on deep-networks (13; 11). This may result from a distribution gap between the TartanAir training set and the test data, however, we exclusively train using TartanAir for fair comparison to other deep-learning approaches.

Additionally, an inherit limitation of sparse VO systems is that they triangulate fewer points, and therefore can only produce sparse 3D reconstructions. Dense reconstructions from DPVO could be obtained by running dense multiview-stereo in a separate process, which is orthogonal to our contribution.

I Network Architecture

Update Operator: In Fig. F, we show the architecture of the update operator, excluding non-learnable layers such as the correlation layer and the bundle adjustment layer. Since the 1D-Convolutions and the Softmax-Aggregation layers operate on neighoring edges, they are partially implemented using PyTorch (7) scattering operations.

Feature Extractor: In Fig. G, we show a visualization of the architecture of the feature extractor. The

same network architecture is used for extracting context features and for extracting matching features, except the former uses no normalization and output dimension $D = 384$ while the latter uses instance normalization and $D = 128$. We use instance normalization for the matching features since they should be calculated independently for each input image in a batch, which other flow-based networks (10; 12; 11) do as well. Our feature extractor is similar to the architecture used in DROID-SLAM (11), but half the size. Consequently, the output resolution is 1/4 of the image resolution, as opposed to 1/8th. Since DPVO only tracks a sparse collection of patches instead of predicting dense flow, we can afford to use higher spatial-resolution features without significant memory overhead.

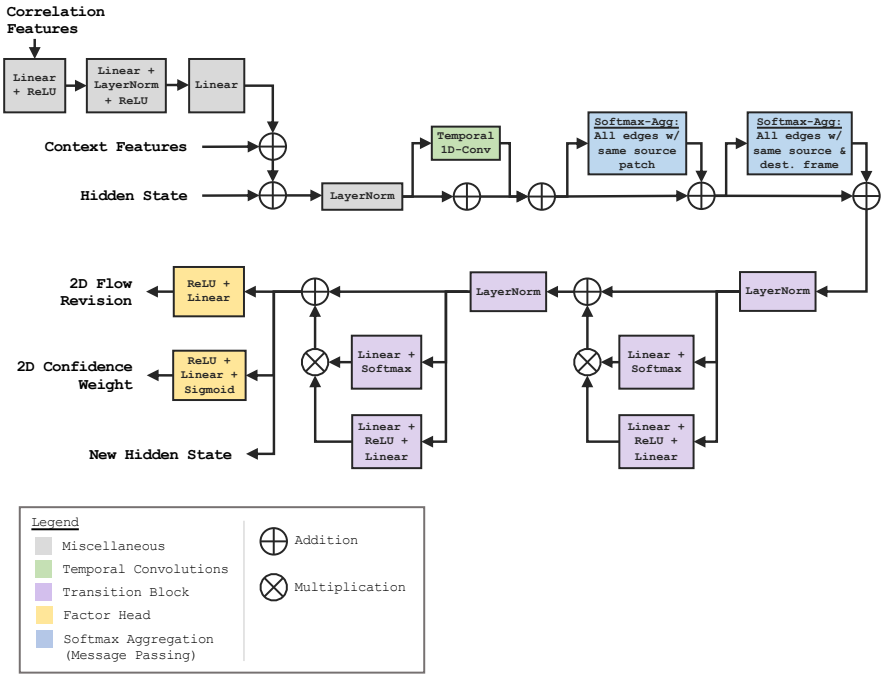


Figure F: Architecture of the Update Operator, excluding non-learnable layers such as the correlation layer and the bundle-adjustment layer.

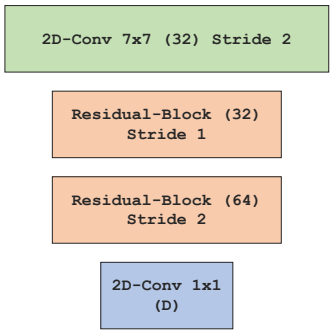


Figure G: Architecture of the feature extractors. $D = 128$ for the matching-feature extractor and $D = 384$ for the context-feature extractor.

J Predicted Trajectories on EuRoC

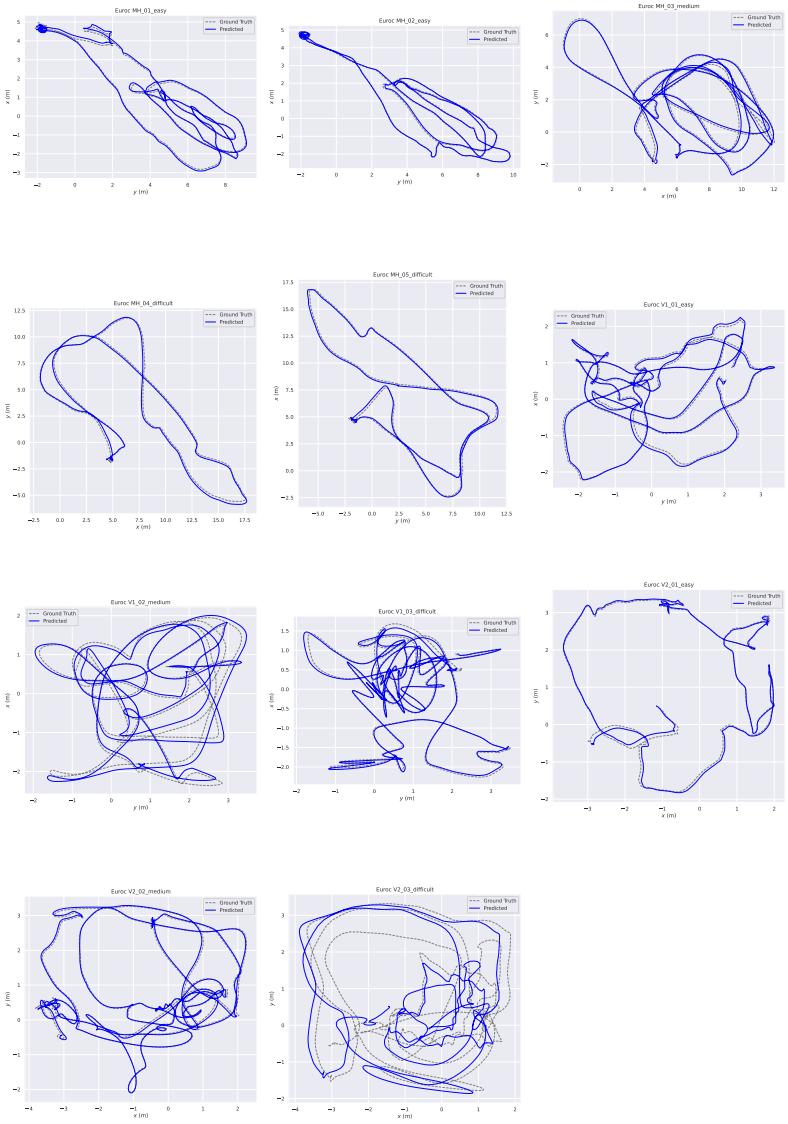


Figure H: Trajectory predictions on the EuRoC test dataset.

K Predicted Trajectories on TartanAir

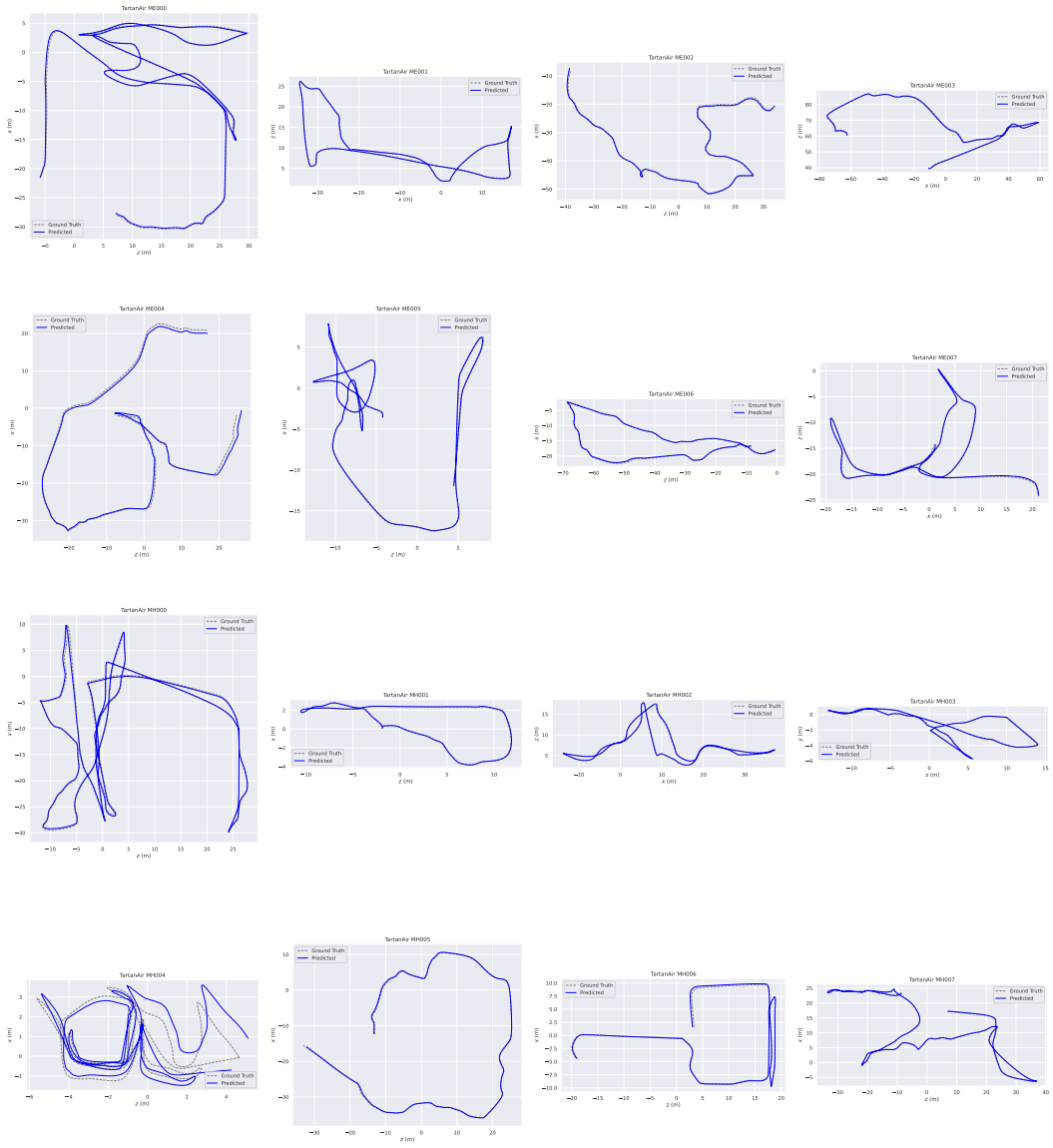


Figure I: Trajectory predictions on the TartanAir test dataset.

References

[1] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[2] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2016.

[3] Michael Grupp. evo: Python package for the evaluation of odometry and slam. `https://github.com/MichaelGrupp/evo`, 2017.

[4] Zhixiang Min and Enrique Dunn. Voldor+ slam: For the times when feature-based or direct methods are not good enough. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13813–13819. IEEE, 2021.

[5] Zhixiang Min, Yiding Yang, and Enrique Dunn. Voldor: Visual odometry from log-logistic dense optical flow residuals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4898–4909, 2020.

[6] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[8] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019.

[9] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle adjusted direct RGB-D SLAM. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[10] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[11] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in Neural Information Processing Systems*, 34:16558–16569, 2021.

[12] Zachary Teed and Jia Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8375–8384, 2021.

[13] Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. Tartanvo: A generalizable learning-based vo. *arXiv preprint arXiv:2011.00359*, 2020.

[14] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916. IEEE, 2020.