APPENDIX for: Learning Interpretable BEV Based VIO without Deep Neural Networks

Anonymous Author(s)

I Pseudocode for DUKF in BEVO

In this Section, We elaborate upon the description of DUFK utilized in BEVO with the pseudocode shown in Algorithm 1, as well as the training process of BEVO in 2. Note: $x.$ stands for state x in time $\cdot$.

II Pseudocode for BEVO+

We further elaborate upon the extension of BEVO as the differentiable front-end of differentiable localization. The localization and the odometry are trained together end to end with the robot's location as the supervision, and is available for localization in heterogeneous maps. Similar to how we retrieve pitch and roll in BEVO, we also utilize the DUKF for localization, and name the whole process as BEVO+. The pseudocode for the localization is shown in Algorithm 3 and Algorithm 4. Note: In these algorithms, $\{x., y., yaw.\}$ stands for the 2D location and the heading angle of the robot in time $\cdot$.

III Experimental Setups for Heterogeneous Localization

We take the GPS data as the ground truth to evaluate the performance of the localization. For each scene, we train the odometry and the localization with the first two quarters of the data, evaluate it with the third quarter, and test it with the last quarter.

- In CARLA, we localize a vehicle on the satellite map in different weathers. We train and utilize BEVO as the odometry and localize the projected BEV (from different weathers) on the heterogeneous satellite map.
- The AeroGround (AG) Dataset is collected for multi-robot collaboration. In this dataset, we train and utilize BEVO as the odometry and localize the ground robot with its front camera BEV on the heterogeneous map built by a drone.

IV Visual Results on Odometery

In this Section, we elaborate upon the visual demonstration of the odometry for sequence 00∼08 of the KITTI dataset. These sequences are the training and validation sets. The demonstrations are shown in Fig. 1. Together with the demonstration of sequence 09∼10, the results show that BEVO stays robust not only in training, validation, but also in the testing. We argue that this is achieved knowing the testing sequences share the same sensor as the training. This proves that the training of BEVO for each sensor can be applied once for all.

V Visual Results on Localization

In this Section, we show more visual results of the differentiable localization, BEVO+. Since the performance of BEVO+ in the real world is demonstrated in the original paper, we gave a set of demonstration in different settings of Carla, to study the robustness of the method, as shown in Fig. 2. We first train the localization in sunny days of Town 1, with randomly generated obstacles,

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

Algorithm 1 Unscented Kalman Filter (UKF)

Input: $x_{t-1}, \Delta x_{t-1}, x\_measure_t$
Output: x_t

Load μ_{t-1}, σ_{t-1} **into this recursion.**
$X_{t-1} \leftarrow$ Sampling$([x_{t-1}], \sigma_{t-1})$
$\bar{X}_t^* \leftarrow$ **MotionModel(**$X_{t-1}, [\Delta x_{t-1}]$ **)**
$\bar{\mu}_t \leftarrow$ WeightedAverage$(\bar{X}_t^*)$
$\bar{\sigma}_t \leftarrow$ WeightedAverage$[(\bar{X}_t^* - \bar{\mu}_t)(\bar{X}_t^* - \bar{\mu}_t)^T]$ + Motion Noise O_t
$\bar{X}_t \leftarrow$ Sampling$(\bar{\mu}_t, \bar{\sigma}_t)$
$\bar{Z}_t \leftarrow$ **MeasurementModel(**$\bar{X}_t$**)**
$\bar{M}_t \leftarrow$ WeightedAverage$(\bar{Z}_t)$
$\bar{\sum}_t \leftarrow$ WeightedAverage$[(\bar{Z}_t - \bar{M}_t)(\bar{Z}_t - \bar{M}_t)^T]$ + Measurement Noise Q_t
$\bar{\sum}_t^{X,Z} \leftarrow$ WeightedAverage$[(\bar{X}_t^i - \bar{\mu}_t)(\bar{Z}_t^i - \bar{M}_t)^T]$
$K_t \leftarrow \bar{\sum}_t^{X,Z} \bar{\sum}_t^{-1}$
$Z_t \leftarrow [x\_measure_t]$
$\mu_t \leftarrow \bar{\mu}_t + K_t(Z_t - \bar{M}_t)$
$\sigma_t \leftarrow \bar{\sigma}_t + K_t \bar{\sum}_t K_t^{-1}$
$x_t \leftarrow \mu_t$
return x_t

Algorithm 2 BEVO

Input: $image_{t-1}, image_t, imu\_data$, Ground Truth:$\mathbf{t}_t^*, \theta_t^*$
Output: 2D translation: $\mathbf{t}_t$, pitch: α_t, roll: β_t, yaw: θ_t

Load $\alpha_{t-1}, \beta_{t-1}$ **from last recursion of BEVO into this recursion.**
Load $\omega_{\alpha_{t-1}}, \omega_{\beta t-1}$ **from** imu_data
Load $acc_t^x, acc_t^y, acc_t^z$ **from** imu_data
Load Δt **from** imu_data
$[\Delta\alpha_{t-1}, \Delta\beta_{t-1}] \leftarrow [\omega_{\alpha_{t-1}}, \omega_{\beta t-1}] \times \Delta t$
$\alpha\_measure_t \leftarrow -arctan(acc_t^x / \sqrt{acc_t^{y\,2} + acc_t^{z\,2}})$
$\beta\_measure_t \leftarrow arctan(acc_t^y / acc_t^z)$
$[\alpha_t, \beta_t] \leftarrow$ **UKF(**$[\alpha_{t-1}, \beta_{t-1}], [\Delta\alpha_{t-1}, \Delta\beta_{t-1}], [\alpha\_measure_t, \beta\_measure_t]$**)**
$image_{t-1}^{bev} \leftarrow$ **BEVProjection(**$image_{t-1}, \alpha_{t-1}, \beta_{t-1}$**)**
$image_t^{bev} \leftarrow$ **BEVProjection(**$image_t, \alpha_t, \beta_t$**)**
$[\mathbf{t}_t, \theta_t] \leftarrow$ **DPC(**$image_{t-1}^b, image_t^b$ **)**
Loss $\mathcal{L}([\mathbf{t}_t^*, \theta_t^*], [\mathbf{t}_t, \theta_t])$
Backward
return $\mathbf{t}_t, \theta_t, \alpha_t, \beta_t$

and test it in Town 2, denoted as "Dynamic Obstacles". Then we remove the dynamic obstacles, also train in Town 1 and test in Town 2, named as "Sunny". Finally, we change the lighting condition to nighttime, and train the localization in Town 1 and test in Town 2, denoted as "Night". The results shows that BEVO+ is robust if the modality of sensors and the global map in the testing stage stay unchanged with that of training stage. Note: The green points which stand for BEVO+ in the figure is almost invisible because they are mostly overlapped with the ground truth.

VI Related Works

In this section, we will introduce the related works of VIO, mainly divided into two parts, traditional methods and learning-based methods.

VI.1 Traditional Methods

Visual-inertial odometry aims to fuse data from the camera and inertial measurement unit to estimate the ego-motion. Traditional VIO methods are mainly based on filtering and optimization. Mourikis

Algorithm 3 BEVO For Localization (BEVO+)

Input: $image_{t-1}, image_t, imu\_data, drone\_map, x_{t-1}, y_{t-1}, yaw_{t-1}$
Output: x_t, y_t, yaw_t
$[\Delta x_{t-1}, \Delta y_{t-1}, \Delta yaw_{t-1}] \leftarrow$ **BEVO(**$image_{t-1}$, $image_t$, imu_data **)**
$[x_t^*, y_t^*, yaw_t^*] \leftarrow [x_{t-1} + \Delta x_{t-1}, y_{t-1} + \Delta y_{t-1}, yaw_{t-1} + \Delta yaw_{t-1}]$
Load bev image of $image_t$ from BEVO as $image_t^b$.
$image_t^* \leftarrow$ **CropInDroneMap(** x_t^*, y_t^*, yaw_t^* **)**
$[\Delta x_t^{'}, \Delta y_t^{'}, \Delta yaw_t^{'}] \leftarrow$ **DPC(** $image_t^b$, $image_t^*$ **)**
$[x_{t\_measure}, y_{t\_measure}, yaw_{t\_measure}] \leftarrow [x_t^* + \Delta x_t^{'}, y_t^* + \Delta y_t^{'}, yaw_t^* + \Delta yaw_t^{'}]$
$[x_t, y_t, yaw_t] \leftarrow$ **UKF_ ForLocalization(**$[x_{t-1}, y_{t-1}, yaw_{t-1}]$,
$[\Delta x_{t-1}, \Delta y_{t-1}, \Delta yaw_{t-1}]$,
$[x_{t\_measure}, y_{t\_measure}, yaw_{t\_measure}]$**)**
return x_t, y_t, yaw_t

Algorithm 4 UKF_ForLocalization

Input: $[x_{t-1}, y_{t-1}, yaw_{t-1}], [\Delta x_{t-1\_odom}, \Delta y_{t-1\_odom}, \Delta yaw_{t-1\_odom}]$,
$[x_{t\_measure}, y_{t\_measure}, yaw_{t\_measure}]$
Output: $[x_t, y_t, yaw_t]$
Load μ_{t-1}, σ_{t-1} **into this recursion.**
$X_{t-1} \leftarrow$ Sampling($[x_{t-1}, y_{t-1}, yaw_{t-1}], \sigma_{t-1}$)
$\bar{X}_t^* \leftarrow$ **MotionModel(**$X_{t-1}, [\Delta x_{t-1\_odom}, \Delta y_{t-1\_odom}, \Delta yaw_{t-1\_odom}]$ **)**
$\bar{\mu}_t \leftarrow$ WeightedAverage($\bar{X}_t^*$)
$\bar{\sigma}_t \leftarrow$ WeightedAverage[$(\bar{X}_t^* - \bar{\mu}_t)(\bar{X}_t^* - \bar{\mu}_t)^T$] + Motion Noise O_t
$\bar{X}_t \leftarrow$ Sampling($\bar{\mu}_t$, $\bar{\sigma}_t$)
$\bar{Z}_t \leftarrow$ **MeasurementModel(**$\bar{X}_t$**)**
$\bar{M}_t \leftarrow$ WeightedAverage($\bar{Z}_t$)
$\bar{\sum}_t \leftarrow$ WeightedAverage[$(\bar{Z}_t - \bar{M}_t)(\bar{Z}_t - \bar{M}_t)^T$] + Measurement Noise Q_t
$\bar{\sum}_t^{X,Z} \leftarrow$ WeightedAverage[$(\bar{X}_t^i - \bar{\mu}_t)(\bar{Z}_t^i - \bar{M}_t)^T$]
$K_t \leftarrow \bar{\sum}_t^{X,Z} \bar{\sum}_t^{-1}$
$Z_t \leftarrow [x_{t\_measure}, y_{t\_measure}, yaw_{t\_measure}]$
$\mu_t \leftarrow \bar{\mu}_t + K_t(Z_t - \bar{M}_t)$
$\sigma_t \leftarrow \bar{\sigma}_t + K_t \sum_t K_t^{-1}$
$[x_t, y_t, yaw_t] \leftarrow \mu_t$
return x_t, y_t, yaw_t

et al. [1] propose a Multi-State Constraint Kalman Filter (MSCKF) method that utilizes the EKF to estimate poses. Moreover, Li et al. [2] improve the MSCKF approach by ensuring the correct observability properties and performing online estimation of calibration parameters. Sun et al. [3] present a stereo version MSCKF which is robust and efficient. OKVIS [4] optimizes through keyframe while VINS-Mono [5] is a state estimator based on nonlinear optimization, which contains a tightly coupled visual-inertial odometry and performs global pose graph optimization. These robust methods can generalize well but require empirical parameter tuning which is labor intensive.

VI.2 Learning-based Methods

VINet [6] is the first end-to-end learning-based method for visual-inertial odometry which eliminates the need for manual synchronization and calibration. DeepVO [7] uses Recurrent Convolutional Neural Networks to learn feature representation in visual odometry problems. Wang et al. [8] present TartanVO, which can generalize to multiple datasets and real-world scenarios. DeepVIO [9] merges 2D optical flow features and IMU data to provide absolute trajectory estimation, during which the depth and dense point cloud are estimated. More recent works, e.g., SelfVIO [10], CodeVIO [11], UnDeepVO [12], Li et al. [13], also take advantage of depth estimation to achieve high pose estimation accuracy. However, all methods above train a large network with millions of parameters, resulting in heavy models and are merely interpretable with weak generalization ability. Therefore, we set to solve this problem by introducing a fully interpretable model with only 4 trainable parameters.

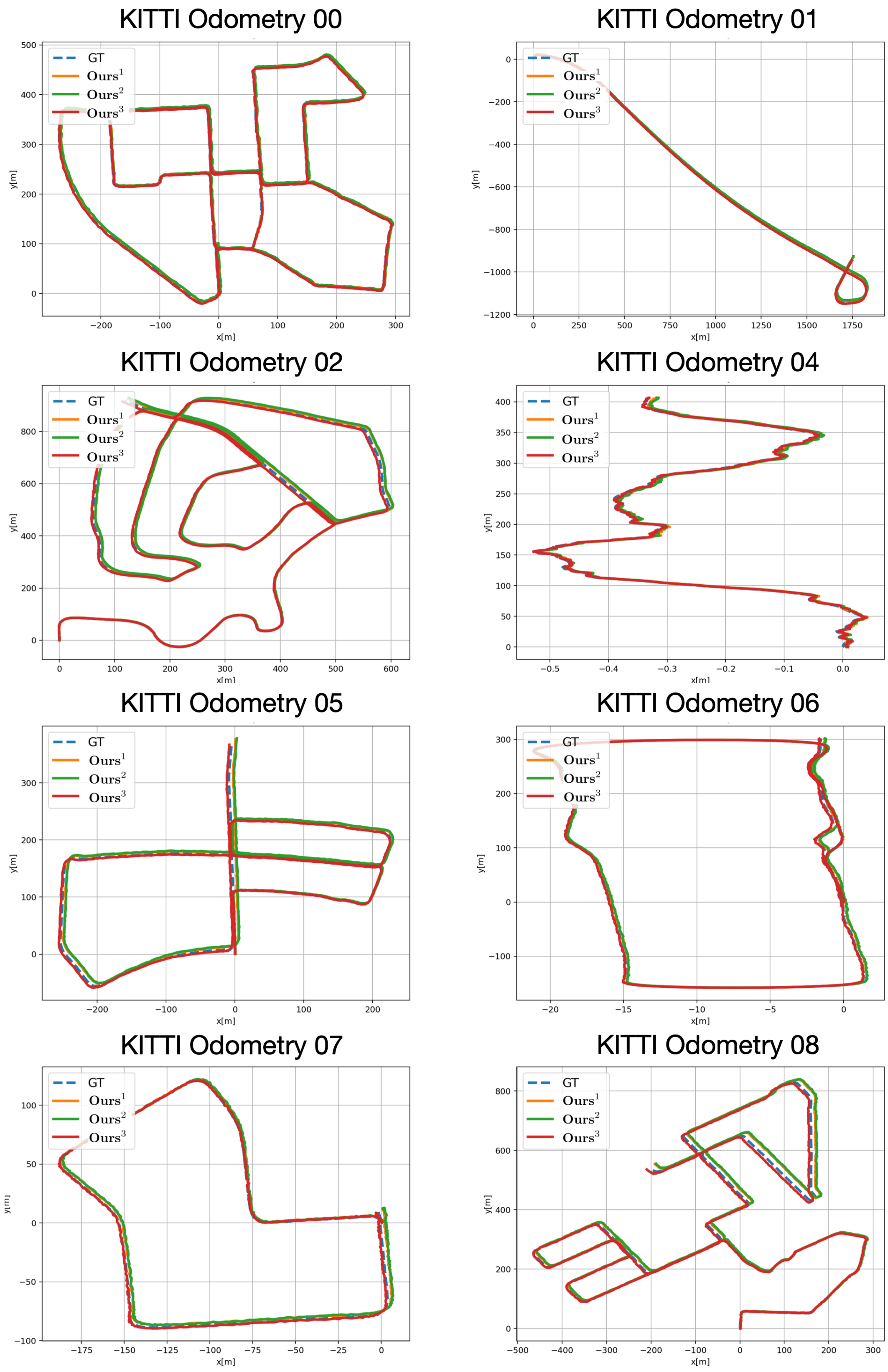

Figure 1: The **visual demonstration** of BEVO in sequence 00∼08 of KITTI.

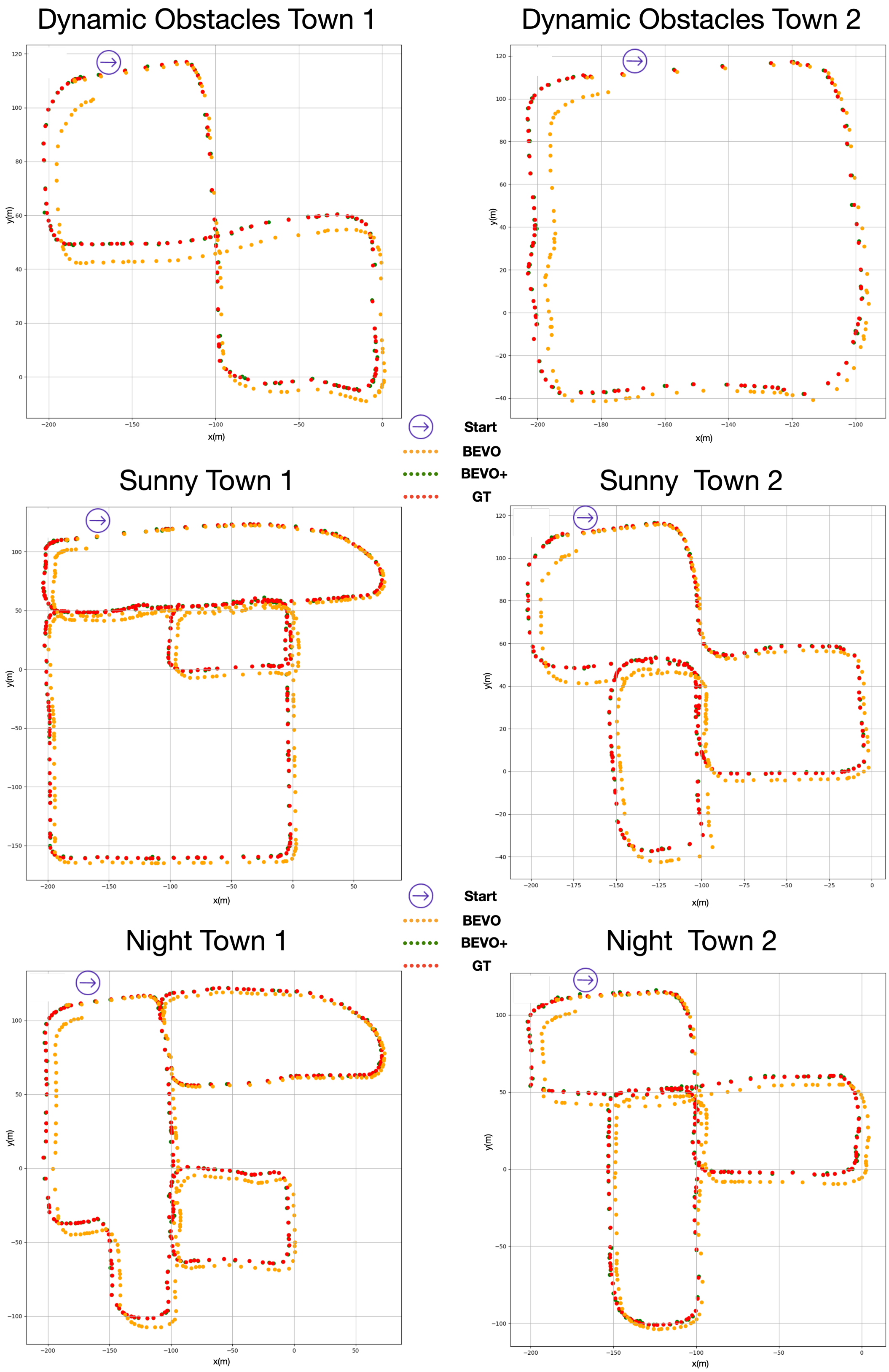

Figure 2: The **qualitative demonstration** of the localization in different conditions of Carla.

References

[1] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572. IEEE, 2007.

[2] M. Li and A. I. Mourikis. High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[3] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, 3(2):965–972, 2018.

[4] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[5] T. Qin, P. Li, and S. Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[6] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[7] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050. IEEE, 2017.

[8] W. Wang, Y. Hu, and S. Scherer. Tartanvo: A generalizable learning-based vo. *arXiv preprint arXiv:2011.00359*, 2020.

[9] L. Han, Y. Lin, G. Du, and S. Lian. Deepvio: Self-supervised deep learning of monocular visual inertial odometry using 3d geometric constraints. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6906–6913. IEEE, 2019.

[10] Y. Almalioglu, M. Turan, A. E. Sari, M. R. U. Saputra, P. P. de Gusmão, A. Markham, and N. Trigoni. Selfvio: Self-supervised deep monocular visual-inertial odometry and depth estimation. *arXiv preprint arXiv:1911.09968*, 2019.

[11] X. Zuo, N. Merrill, W. Li, Y. Liu, M. Pollefeys, and G. Huang. Codevio: Visual-inertial odometry with learned optimizable dense depth. *arXiv preprint arXiv:2012.10133*, 2020.

[12] R. Li, S. Wang, Z. Long, and D. Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 7286–7291. IEEE, 2018.

[13] S. Li, X. Wang, Y. Cao, F. Xue, Z. Yan, and H. Zha. Self-supervised deep visual odometry with online adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6339–6348, 2020.