Monocular Visual-Inertial Odometry

Thomas Mörwald[1]

1. Overview

The proposed algorithm is an optimization-based approach in a fixed-lag, i.e. sliding window setting. It is causal, without any post processing or batch optimization of the whole trajectory. No loop closure nor lost key point retrieval is used. We are only using measurements from the IMU and the left image of the Qualcomm Snapdragon flight board.

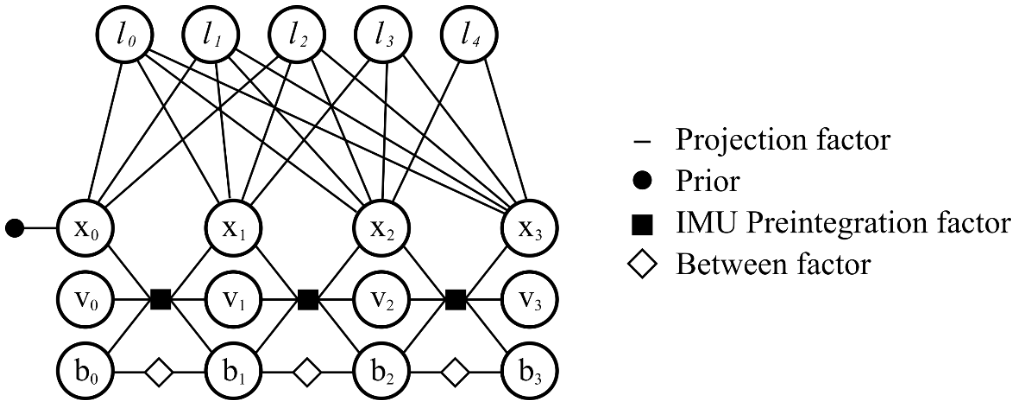

Figure 1: *Factor graph with all variables and factors involved. The state of the respective image frames consists of pose* ***x****, velocity* ***v*** *and biases for accelerometer and gyroscope* ***b****. The states are constrained by IMU pre-integration factors. Additionally, landmarks are required only to vary within the boundaries of the random walk values given in the specifications of the IMU (i.e. provided calibration of the datasets). Landmarks are denoted by* ***l*** *and are observed in the images via projection factors*

2. Initialization

As stated in *Claim 4* of [4] successful initialization can only be achieved when the following properties are fulfilled. First, the IMU must be significantly excited, i.e. the variation of acceleration exceeds a certain threshold. Second, enough translational movement must take place to render the landmarks observable by the image sequence. And third, the states are only observable in a local coordinate frame.

Inspired by [1] we first use only visual measurements to initialize poses and landmarks. Afterwards, we align the resulting trajectory to the inertial measurements, estimating scale, gravity and all velocities involved. The initial condition $\boldsymbol{x}_0 = [\boldsymbol{R}, \boldsymbol{t}] = [\boldsymbol{I}, \boldsymbol{0}]$ is enforced. The biases for the accelerometer and gyroscope are initialized as zero and estimated accurately within the first full optimization as described in Section 4.

3. Image processing frontend

To obtain visual measurements we use the "Good features to track" algorithm [5] as implemented in OpenCV. We try to maintain a fixed number of key points, spread more-or-less homogenously throughout the image. Therefore, we require the detector only to include key points that are a minimum distance apart from any other key points.

For finding corresponding key points in the subsequent images, we employ the well-known pyramidal implementation of the Lucas-Kanade optical flow [6].

4. Optimization backend

Key to any robust Visual-Inertial Odometry (VIO) algorithm is a well-designed sensor fusion backend. In recent years optimization-based approaches became very popular. The reason for this is that the mathematical formulation of VIO as well as Visual SLAM has been better understood, which lead to

[1] Thomas Mörwald (PhD) is Senior Engineer R&D and Computer Vision Expert at Leica Geosystems, part of Hexagon.

several software packages that are specifically designed to exploit the structure of the problem (e.g. sparsity and block diagonality). Our approach relies on Incremental Smoothing and Mapping, iSAM2 [2], implemented within the library GTSAM (*https://github.com/borglab/gtsam*).

Further we use pre-integration factors from [3] to avoid re-integration of IMU measurements whenever the state changes. As shown in **Figure 1**, a state consists of a pose $\boldsymbol{x}$, velocity $\boldsymbol{v}$ and the biases for accelerometer and gyroscope $\boldsymbol{b}$. Landmarks are denoted by $\boldsymbol{l}$ and observed via projection factors, parametrizing them as 3d coordinates in space. New landmarks are initialized by triangulation whenever its parallax in the image exceeds a certain threshold. For this purpose, we compensate for the rotational part of the optical flow.

For real-time processing performance we use the fixed-lag implementation of GTSAM, which efficiently marginalizes out states that are older than a certain time window. We used a fixed-lag of 0.5 seconds to obtain the results from **Table 1** and **Table 2**.

5. Experiments

For processing we use an HP laptop with an Intel® Core ™ i7-8650U CPU @ 1.9 GHz and 32 GB RAM. No GPU was used. The algorithm was executed on Ubuntu 18.04.

All sequences were executed with the same set of parameters.

The total processing time for each sequence is shown in **Table 1**. To demonstrate the real-time capability of the approach we also present the processing times per frame in **Table 2**.

Environment Camera view Dataset #	indoor 45 3	indoor 45 16	indoor forward 11	indoor forward 12	outdoor forward 9	outdoor forward 10
Marginalization	3.3	1.5	3.8	2.6	3.5	3.9
Optimization	32.9	14.3	27.5	14.8	25.7	33.0
Outlier removal	1.5	0.6	1.5	0.9	1.6	1.9
Triangulation	0.7	0.4	1.0	0.7	1.0	1.2
Key-point detection	19.7	12.3	19.3	11.8	18.4	23.7
Optical Flow	8.7	6.2	8.8	6.9	13.5	18.0
Total [s]	**66.7**	**35.4**	**61.8**	**37.7**	**63.7**	**81.8**

Table 1: *Total processing times in seconds (s) for each sequence.*

Environment Camera view Dataset #	indoor 45 3	indoor 45 16	indoor forward 11	indoor forward 12	outdoor forward 9	outdoor forward 10
Marginalization	1.7	1.4	1.9	1.8	1.6	1.4
Optimization	17.4	13.0	13.3	10.3	11.9	12.1
Outlier removal	0.8	0.6	0.7	0.7	0.7	0.7
Triangulation	0.4	0.4	0.5	0.5	0.5	0.4
Key-point detection	9.5	9.5	8.7	7.2	7.8	8.1
Optical Flow	4.2	4.8	4.0	4.2	5.7	6.2
Total [ms]	**33.9**	**29.6**	**29.0**	**24.6**	**28.2**	**29.0**

Table 2: *Per frame processing times in milliseconds (ms) for each sequence.*

References

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

[2] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., & Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, *31*(2), 216-235.

[3] Forster, C., Carlone, L., Dellaert, F., & Scaramuzza, D. (2016). On-Manifold Preintegration for Real-Time Visual--Inertial Odometry. *IEEE Transactions on Robotics*, *33*(1), 1-21.

[4] Jones, E. S., & Soatto, S. (2011). Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research*, *30*(4), 407-430.

[5] Shi, J. (1994, June). Good features to track. In 1994 Proceedings of IEEE conference on computer vision and pattern recognition (pp. 593-600). IEEE.

[6] Bouguet, J. Y. (2001). Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, *5*(1-10), 4.