Visual-Inertial Navigation Systems: An Introduction

Patrick Geneva

University of Delaware

May 31, 2021

Outline

- Introduction
- Visual-Inertial Estimators
- Essential Building Blocks
- Open Sourced Systems, Datasets, and Evaluation
- Discussion of Future Directions
- Conclusion

Motivation

- Robotics present challenging environments for **realizing** estimation, perception, and SLAM!
- Visual-inertial navigation systems (VINS) are **crucial** and **well suited** for many applications

Extraterrestrial Robots

Autonomous Driving

Wearables and Health Tracking

AR / VR Experiences

Micro Aerial Vehicles

Human Pose Tracking

Warehouse Robotics

Nano Aerial Vehicles

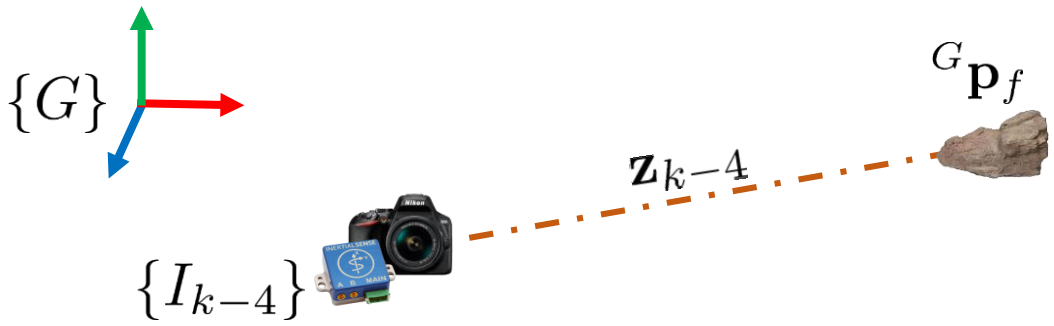
Significance of VINS

- Improving the ***computational efficiency, robustness,*** and ***accuracy*** are challenging open research problems
- Improvements directly impact all applications which leverage VINS
- Directly allow for:
    1. Low-cost: sensors, compute platforms, and robots
    2. Reduced computational energy (longer missions)
    3. Better robustness and accuracy opens gateway for more demanding scenarios

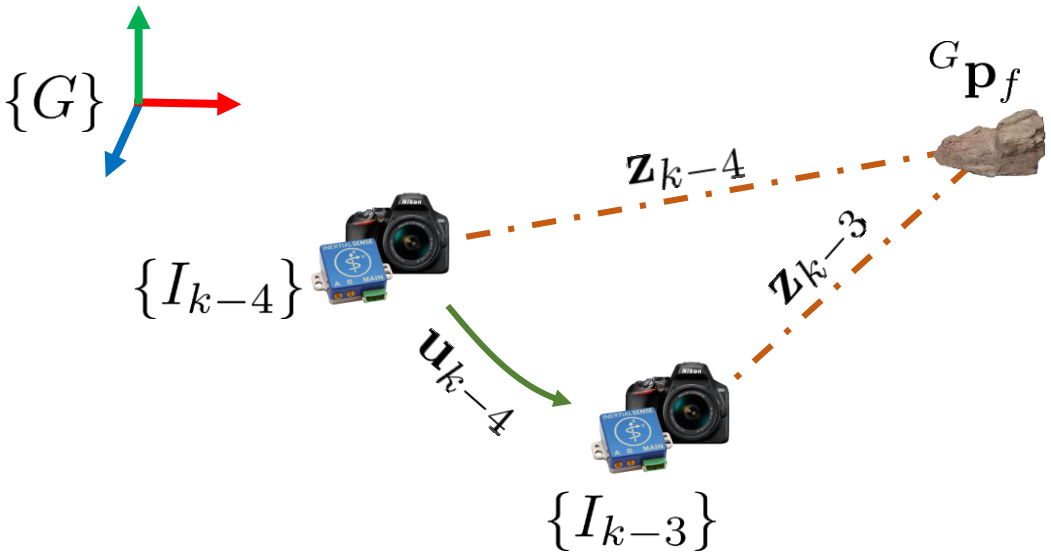
Outline - Estimation Methodology

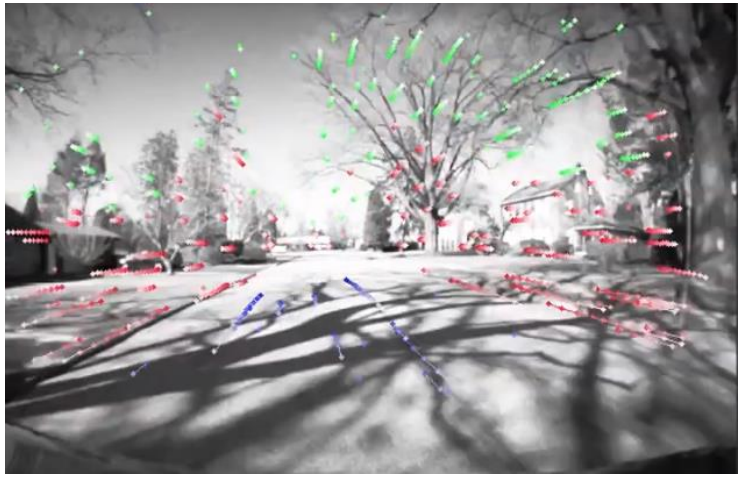
- Multi-State Constraint Kalman Filter (MSCKF)
- Batch Least Squares (BLS) with Pre-integration
- Incremental Optimization (square-root information form)

Problem: Visual-Inertial Estimation

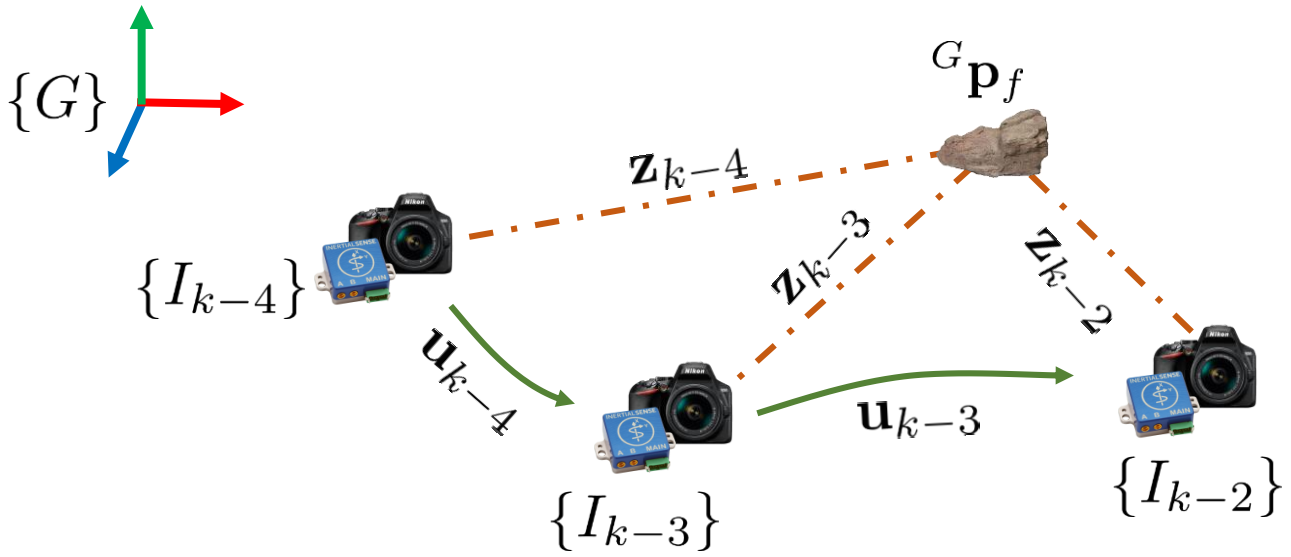


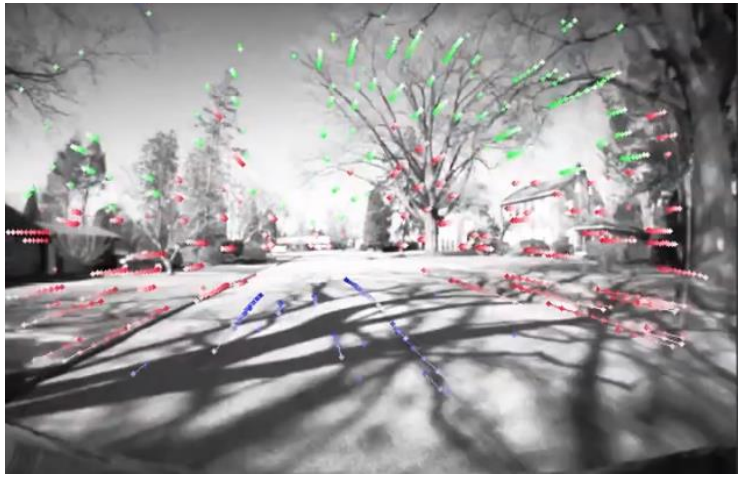
Problem: Visual-Inertial Estimation



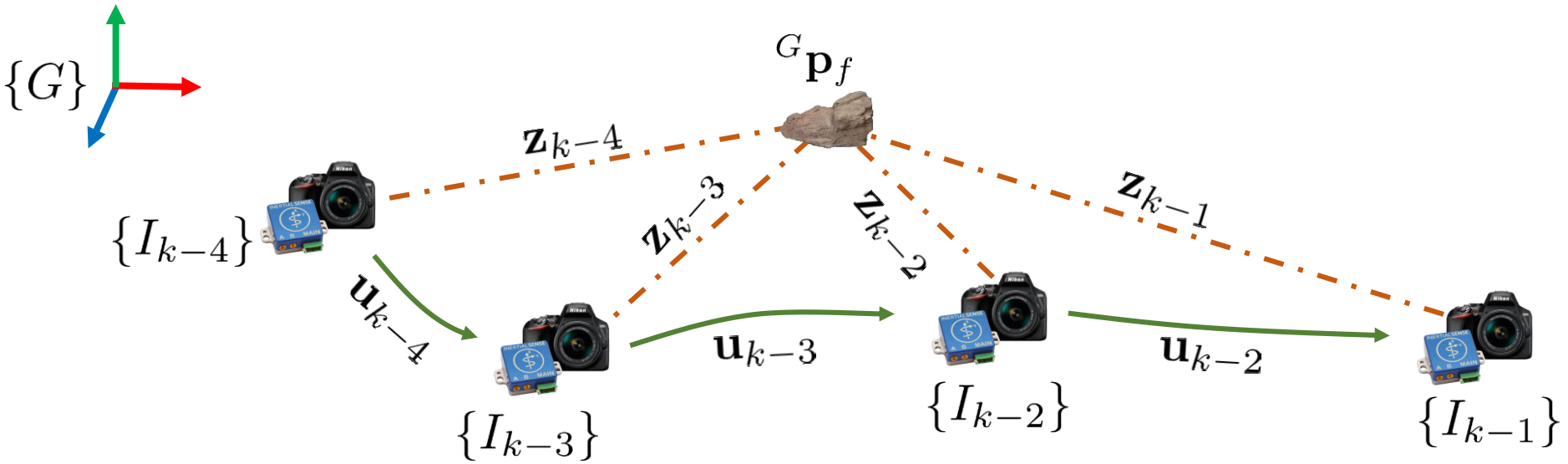


Problem: Visual-Inertial Estimation

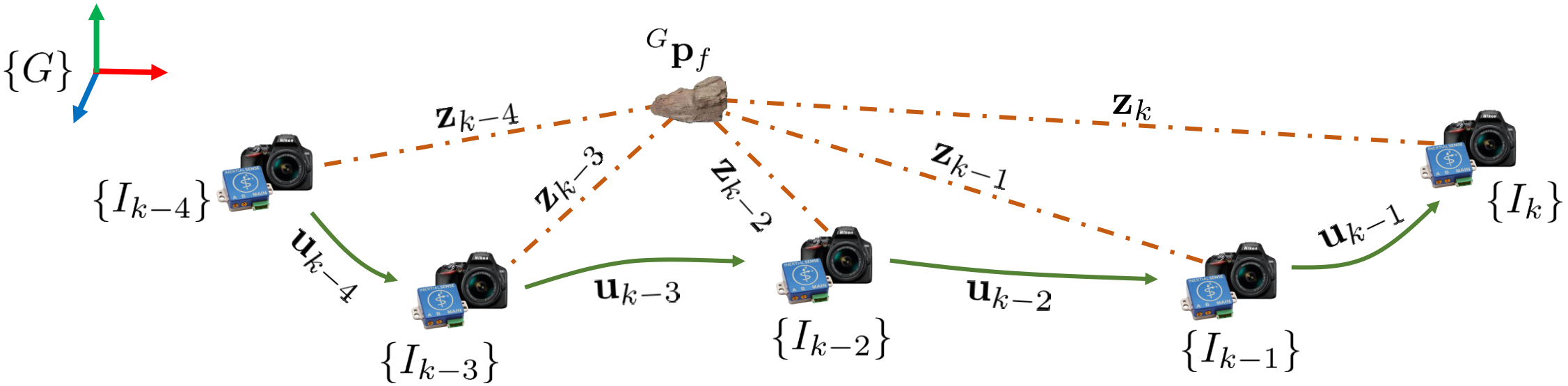




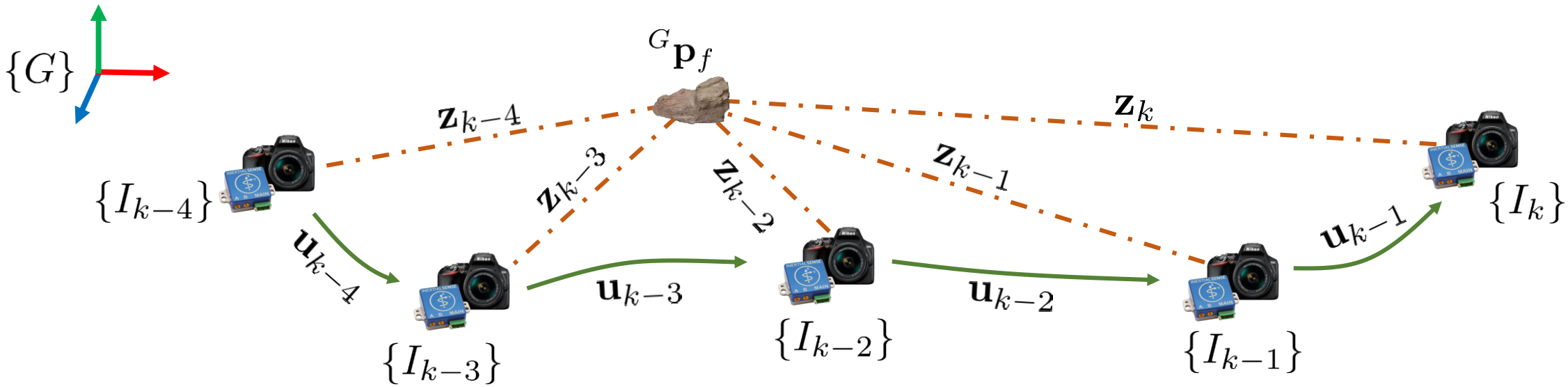
Problem: Visual-Inertial Estimation



Problem: Visual-Inertial Estimation



Problem: Visual-Inertial Estimation



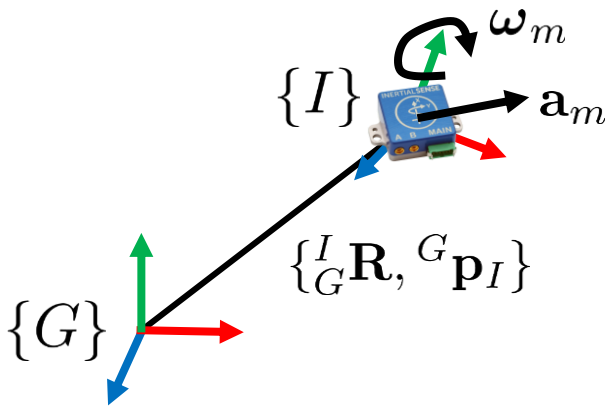
- Given: Bearings $\mathbf{z}_{k-4:k}$ and inertial readings $\mathbf{u}_{k-4:k}$
- Goal: Estimate inertial states and features $\mathbf{x}_k = \begin{bmatrix}\mathbf{x}_I^\top & \mathbf{x}_C^\top & \mathbf{x}_f^\top\end{bmatrix}^\top$

$$\mathbf{x}_I = \begin{bmatrix}{}^{I_k}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_k}^\top & {}^G\mathbf{v}_{I_k}^\top & \mathbf{b}_{\omega_k}^\top & \mathbf{b}_{a_k}^\top\end{bmatrix}^\top$$
$$\mathbf{x}_C = \begin{bmatrix}{}^{I_{k-1}}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_{k-1}}^\top & \cdots & {}^{I_{k-4}}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_{k-4}}^\top\end{bmatrix}^\top$$
$$\mathbf{x}_f = \begin{bmatrix}{}^G\mathbf{p}_f^\top\end{bmatrix}^\top$$

Models: Inertial Kinematics

IMU Measurement Model:

- Gyroscope: $\boldsymbol{\omega}_m = \boldsymbol{\omega} + \mathbf{b}_\omega + \mathbf{n}_\omega$
- Accelerometer: $\mathbf{a}_m = \mathbf{a} + {}^I_G\mathbf{R}\,{}^G\mathbf{g} + \mathbf{b}_a + \mathbf{n}_a$

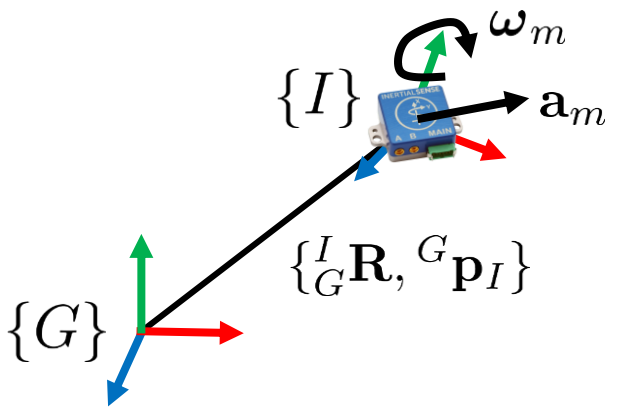


$\bar{q}$	: quaternion
$\mathbf{R}$	: rotation matrix
$\mathbf{p}$	: position
$\mathbf{v}$	: velocity
$\boldsymbol{\omega}$	: angular velocity
$\mathbf{a}$	: linear acceleration
$\mathbf{b}_\omega$	: gyro. bias
$\mathbf{b}_a$	: accel. bias
$\mathbf{g}$	: global gravity
$\mathbf{n}_\omega$	: gyro white noise
$\mathbf{n}_{w\omega}$	: gyro random walk
$\mathbf{n}_a$	: accel white noise
$\mathbf{n}_{wa}$	: accel random walk

Models: Inertial Kinematics

IMU Measurement Model:

- Gyroscope: $\boldsymbol{\omega}_m = \boldsymbol{\omega} + \mathbf{b}_\omega + \mathbf{n}_\omega$
- Accelerometer: $\mathbf{a}_m = \mathbf{a} + {}^I_G\mathbf{R}\,{}^G\mathbf{g} + \mathbf{b}_a + \mathbf{n}_a$



State Evolution Model:

$$
\begin{aligned}
{}^I_G\dot{\bar{q}}(t) &= \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_m(t) - \mathbf{b}_\omega(t) - \mathbf{n}_\omega(t)){}^I_G\bar{q}(t) \\
{}^G\dot{\mathbf{p}}_I(t) &= {}^G\mathbf{v}_I(t) \\
{}^G\dot{\mathbf{v}}_I(t) &= \mathbf{R}({}^I_G\bar{q}(t))^\top(\mathbf{a}_m(t) - \mathbf{b}_a(t) - \mathbf{n}_a(t)) - {}^G\mathbf{g} \\
\dot{\mathbf{b}}_\mathbf{g}(t) &= \mathbf{n}_{w\omega} \\
\dot{\mathbf{b}}_\mathbf{a}(t) &= \mathbf{n}_{wa}
\end{aligned}
$$

Continuous-Time State Evolution Model

$$\dot{\mathbf{x}}_I(t) = \mathbf{f}_c(\mathbf{x}_I(t), \mathbf{u}(t), \mathbf{n}(t))$$

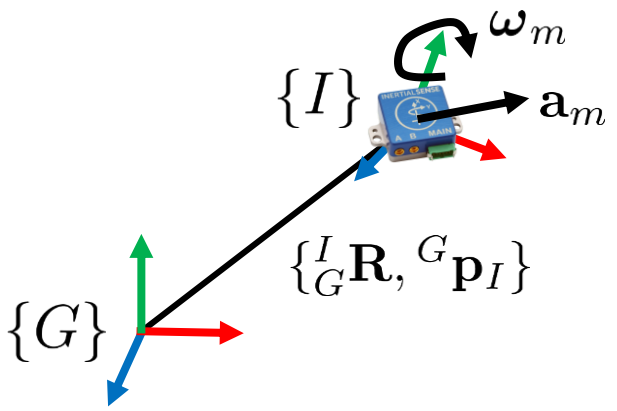
$$\mathbf{u}(t) = \begin{bmatrix}\boldsymbol{\omega}_m(t) & \mathbf{a}_m(t)\end{bmatrix}^\top$$

$\bar{q}$	: quaternion
$\mathbf{R}$	: rotation matrix
$\mathbf{p}$	: position
$\mathbf{v}$	: velocity
$\boldsymbol{\omega}$	: angular velocity
$\mathbf{a}$	: linear acceleration
$\mathbf{b}_\omega$	: gyro. bias
$\mathbf{b}_a$	: accel. bias
$\mathbf{g}$	: global gravity
$\mathbf{n}_\omega$	: gyro white noise
$\mathbf{n}_{w\omega}$	: gyro random walk
$\mathbf{n}_a$	: accel white noise
$\mathbf{n}_{wa}$	: accel random walk

Models: Inertial Kinematics

IMU Measurement Model:

- Gyroscope: $\boldsymbol{\omega}_m = \boldsymbol{\omega} + \mathbf{b}_\omega + \mathbf{n}_\omega$
- Accelerometer: $\mathbf{a}_m = \mathbf{a} + {}^I_G\mathbf{R}\,{}^G\mathbf{g} + \mathbf{b}_a + \mathbf{n}_a$



State Evolution Model:

$$
\begin{aligned}
{}^I_G\dot{\bar{q}}(t) &= \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_m(t) - \mathbf{b}_\omega(t) - \mathbf{n}_\omega(t)){}^I_G\bar{q}(t) \\
{}^G\dot{\mathbf{p}}_I(t) &= {}^G\mathbf{v}_I(t) \\
{}^G\dot{\mathbf{v}}_I(t) &= \mathbf{R}({}^I_G\bar{q}(t))^\top(\mathbf{a}_m(t) - \mathbf{b}_a(t) - \mathbf{n}_a(t)) - {}^G\mathbf{g} \\
\dot{\mathbf{b}}_\mathbf{g}(t) &= \mathbf{n}_{w\omega} \\
\dot{\mathbf{b}}_\mathbf{a}(t) &= \mathbf{n}_{wa}
\end{aligned}
$$

Continuous-Time State Evolution Model

$$\dot{\mathbf{x}}_I(t) = \mathbf{f}_c(\mathbf{x}_I(t), \mathbf{u}(t), \mathbf{n}(t))$$

$$\mathbf{u}(t) = \begin{bmatrix}\boldsymbol{\omega}_m(t) & \mathbf{a}_m(t)\end{bmatrix}^\top$$

Discrete State Evolution Model

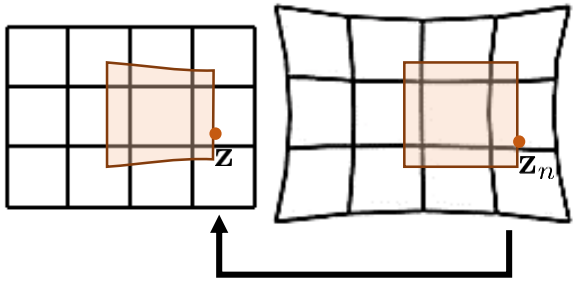
$$\mathbf{x}_k = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$$

$\bar{q}$	: quaternion
$\mathbf{R}$	: rotation matrix
$\mathbf{p}$	: position
$\mathbf{v}$	: velocity
$\boldsymbol{\omega}$	: angular velocity
$\mathbf{a}$	: linear acceleration
$\mathbf{b}_\omega$	: gyro. bias
$\mathbf{b}_a$	: accel. bias
$\mathbf{g}$	: global gravity
$\mathbf{n}_\omega$	: gyro white noise
$\mathbf{n}_{w\omega}$	: gyro random walk
$\mathbf{n}_a$	: accel white noise
$\mathbf{n}_{wa}$	: accel random walk

Models: Camera Measurements

- Distort to “raw” uv from the “ideal” image plane uv

$$\mathbf{z}_m = \mathbf{h}_d(\mathbf{z}_n,\ \boldsymbol{\zeta}) + \mathbf{n}_{pix}$$



[1] Geneva, Patrick, et al. "OpenVINS: A research platform for visual-inertial estimation." 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020.

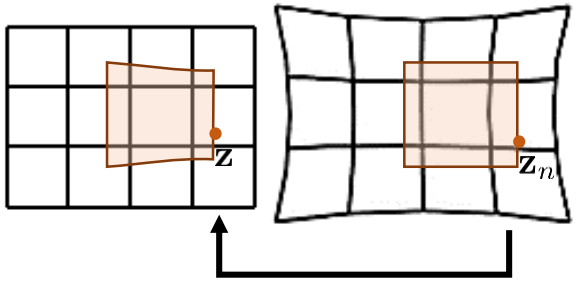
Models: Camera Measurements

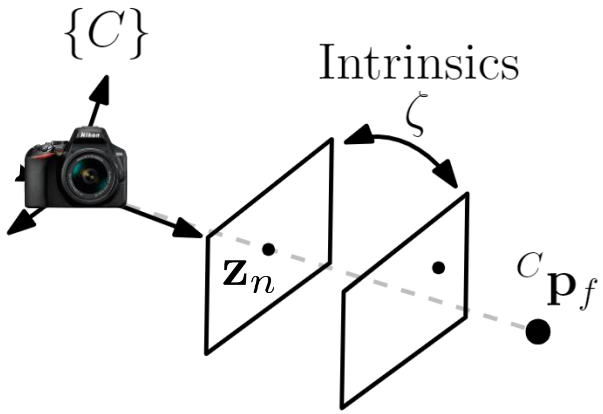
- Distort to “raw” uv from the “ideal” image plane uv

$$\mathbf{z}_m = \mathbf{h}_d(\mathbf{z}_n,\ \boldsymbol{\zeta}) + \mathbf{n}_{pix}$$

- Project onto the “ideal” image plane

$$\mathbf{z}_n = \mathbf{h}_p({}^C\mathbf{p}_f) = \begin{bmatrix} {}^Cx_f / {}^Cz_f \\ {}^Cy_f / {}^Cz_f \end{bmatrix}$$





[1] Geneva, Patrick, et al. "OpenVINS: A research platform for visual-inertial estimation." 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020.

Models: Camera Measurements

- Distort to “raw” uv from the “ideal” image plane uv

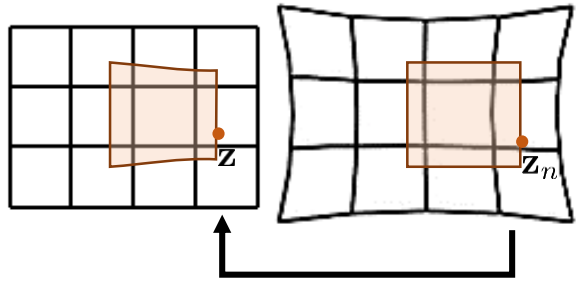
$$\mathbf{z}_m = \mathbf{h}_d(\mathbf{z}_n,\ \boldsymbol{\zeta}) + \mathbf{n}_{pix}$$

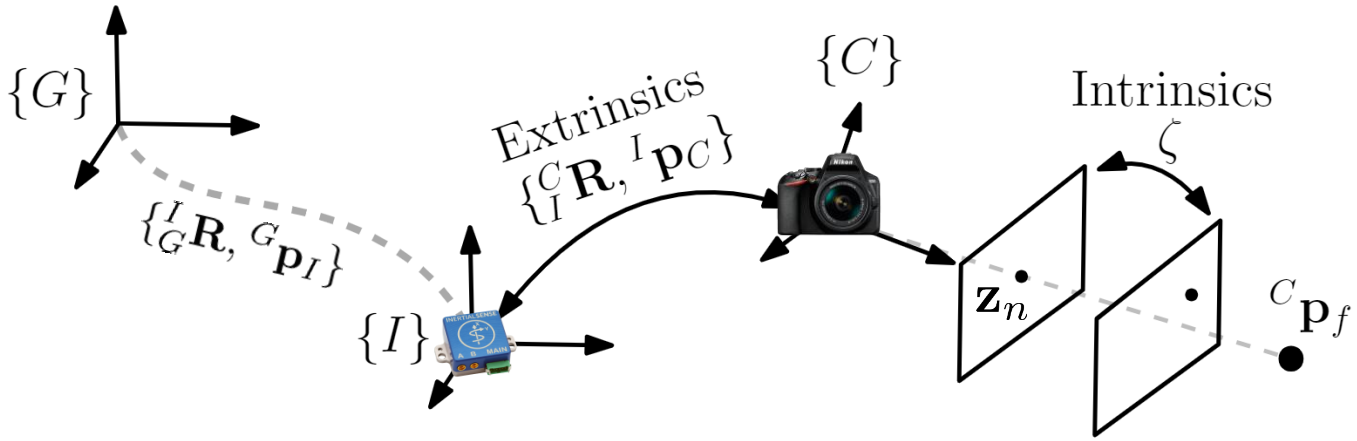
- Project onto the “ideal” image plane

$$\mathbf{z}_n = \mathbf{h}_p({}^C\mathbf{p}_f) = \begin{bmatrix} {}^Cx_f/{}^Cz_f \\ {}^Cy_f/{}^Cz_f \end{bmatrix}$$

- From global ${}^G\mathbf{p}_f$ to camera frame ${}^C\mathbf{p}_f$

$$\begin{aligned} {}^C\mathbf{p}_f &= \mathbf{h}_t({}^G\mathbf{p}_f,\ {}^I_G\mathbf{R},\ {}^G\mathbf{p}_I,\ {}^C_I\mathbf{R},\ {}^C\mathbf{p}_I) \\ &= {}^C_I\mathbf{R}\underbrace{{}^I_G\mathbf{R}({}^G\mathbf{p}_f - {}^G\mathbf{p}_I)}_{\text{G to I transformation}} + {}^C\mathbf{p}_I \end{aligned}$$





[1] Geneva, Patrick, et al. "OpenVINS: A research platform for visual-inertial estimation." 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020.

Models: Camera Measurements

- Distort to “raw” uv from the “ideal” image plane uv

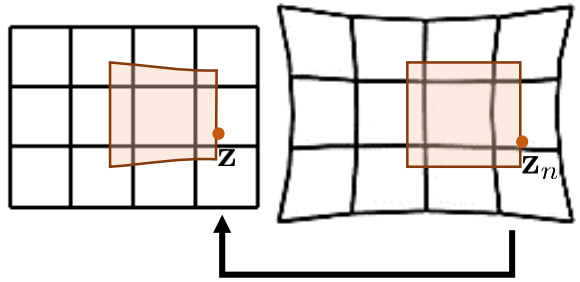
$$\mathbf{z}_m = \mathbf{h}_d(\mathbf{z}_n,\ \boldsymbol{\zeta}) + \mathbf{n}_{pix}$$

- Project onto the “ideal” image plane

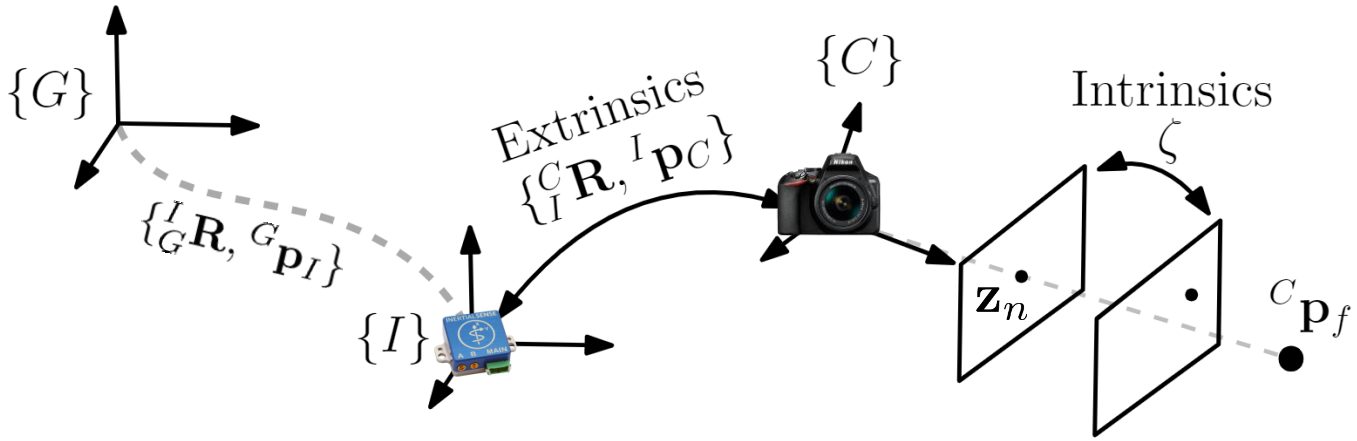
$$\mathbf{z}_n = \mathbf{h}_p({}^C\mathbf{p}_f) = \begin{bmatrix} {}^Cx_f/{}^Cz_f \\ {}^Cy_f/{}^Cz_f \end{bmatrix}$$

- From global ${}^G\mathbf{p}_f$ to camera frame ${}^C\mathbf{p}_f$

$$\begin{aligned} {}^C\mathbf{p}_f &= \mathbf{h}_t({}^G\mathbf{p}_f,\ {}^I_G\mathbf{R},\ {}^G\mathbf{p}_I,\ {}^C_I\mathbf{R},\ {}^C\mathbf{p}_I) \\ &= {}^C_I\mathbf{R}\underbrace{{}^I_G\mathbf{R}({}^G\mathbf{p}_f - {}^G\mathbf{p}_I)}_{\text{G to I transformation}} + {}^C\mathbf{p}_I \end{aligned}$$



distort $\mathbf{z}_n$ to $\mathbf{z}$ in the raw image



Fully combined feature measurement function:

$$\begin{aligned} \mathbf{z}_m &= \mathbf{h}(\mathbf{x}_k) + \mathbf{n}_{pix} \\ &= \mathbf{h}_d(\mathbf{h}_p(\mathbf{h}_t({}^G\mathbf{p}_f,\ {}^I_G\mathbf{R},\ {}^G\mathbf{p}_I,\ {}^C_I\mathbf{R},\ {}^C\mathbf{p}_I)),\ \boldsymbol{\zeta}) + \mathbf{n}_{pix} \end{aligned}$$

[1] Geneva, Patrick, et al. "OpenVINS: A research platform for visual-inertial estimation." 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020.

Background - Extended Kalman Filter (EKF)

Propagation

$$\mathbf{u}_{k-1} = \begin{bmatrix} \boldsymbol{\omega}_{m,k-1} & \mathbf{a}_{m,k-1} \end{bmatrix}^\top$$

$$\mathbf{x}_{k-1} \sim \mathcal{N}(\hat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1}) \quad \mathbf{x}_k = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$$

prior — control input / state evolution

$$\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k^{\ominus}, \mathbf{P}_k^{\ominus})$$

Background - Extended Kalman Filter (EKF)

Propagation

$$\mathbf{u}_{k-1} = \begin{bmatrix} \boldsymbol{\omega}_{m,k-1} & \mathbf{a}_{m,k-1} \end{bmatrix}^\top$$

$$\mathbf{x}_{k-1} \sim \mathcal{N}(\hat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1}) \quad \mathbf{x}_k = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$$

prior — control input / state evolution

$$\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k^{\ominus}, \mathbf{P}_k^{\ominus})$$

Update

$$\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k^{\ominus}, \mathbf{P}_k^{\ominus}) \qquad \mathbf{z}_m = \mathbf{h}(\mathbf{x}_k) + \mathbf{n}$$

propagation state — observation

$$\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k^{\oplus}, \mathbf{P}_k^{\oplus})$$

Background - Extended Kalman Filter (EKF)

Propagation

$$\mathbf{u}_{k-1} = \begin{bmatrix} \boldsymbol{\omega}_{m,k-1} & \mathbf{a}_{m,k-1} \end{bmatrix}^\top$$

$$\mathbf{x}_{k-1} \sim \mathcal{N}(\hat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1}) \quad \mathbf{x}_k = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$$

prior — control input / state evolution

$$\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k^{\ominus}, \mathbf{P}_k^{\ominus})$$

Update

$$\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k^{\ominus}, \mathbf{P}_k^{\ominus}) \qquad \mathbf{z}_m = \mathbf{h}(\mathbf{x}_k) + \mathbf{n}$$

propagation state — observation

$$\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k^{\oplus}, \mathbf{P}_k^{\oplus})$$

Linearization:

$$\mathbf{z}_m = \mathbf{h}(\hat{\mathbf{x}}_k^{\ominus}) + \mathbf{H}_k\tilde{\mathbf{x}} + \mathbf{n}$$

$$\mathbf{r}_k = \mathbf{H}_k\tilde{\mathbf{x}} + \mathbf{n}$$

$$\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$$

Background - Extended Kalman Filter (EKF)

Propagation

$$\mathbf{u}_{k-1} = \begin{bmatrix} \boldsymbol{\omega}_{m,k-1} & \mathbf{a}_{m,k-1} \end{bmatrix}^\top$$

$$\mathbf{x}_{k-1} \sim \mathcal{N}(\hat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1}) \quad \mathbf{x}_k = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$$

prior — control input / state evolution

$$\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k^{\ominus}, \mathbf{P}_k^{\ominus})$$

Update

$$\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k^{\ominus}, \mathbf{P}_k^{\ominus}) \qquad \mathbf{z}_m = \mathbf{h}(\mathbf{x}_k) + \mathbf{n}$$

propagation state — observation

$$\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k^{\oplus}, \mathbf{P}_k^{\oplus})$$

Standard EKF Update

state: $\hat{\mathbf{x}}_k^{\oplus} = \hat{\mathbf{x}}_k^{\ominus} + \mathbf{K}_k \mathbf{r}$

covariance: $\mathbf{P}_k^{\oplus} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^{\ominus}$

where:

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^{\ominus} \mathbf{H}_k^\top + \mathbf{R}_k$$

$$\mathbf{K}_k = \mathbf{P}_k^{\ominus} \mathbf{H}_k^\top \mathbf{S}_k^{-1}$$

$$\mathbf{r}_k = \mathbf{z}_m - \mathbf{h}(\hat{\mathbf{x}}_k^{\ominus})$$

Linearization:

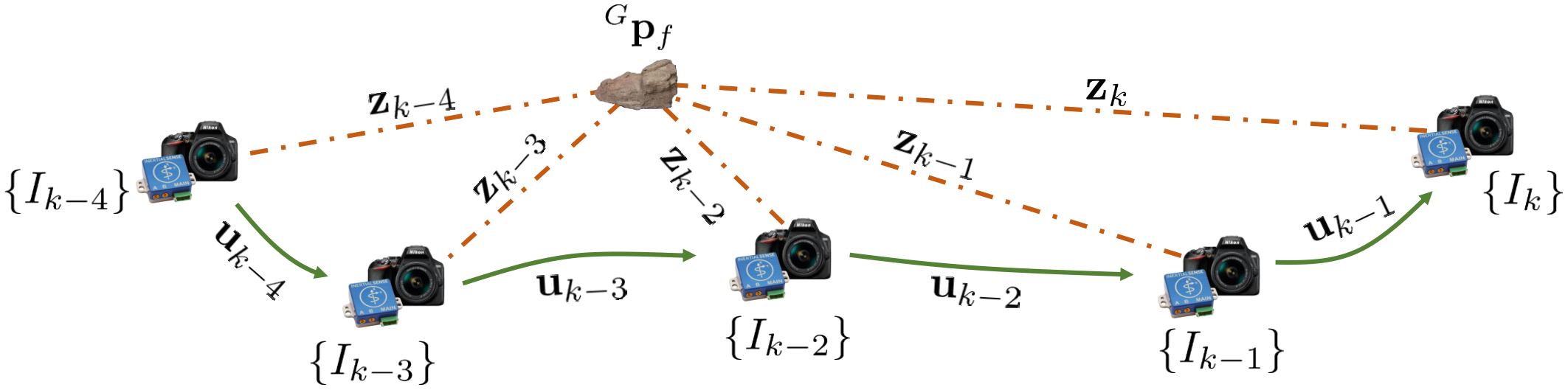
$$\mathbf{z}_m = \mathbf{h}(\hat{\mathbf{x}}_k^{\ominus}) + \mathbf{H}_k \tilde{\mathbf{x}} + \mathbf{n}$$

$$\mathbf{r}_k = \mathbf{H}_k \tilde{\mathbf{x}} + \mathbf{n}$$

$$\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$$

Multi-State Constraint Kalman Filter (MSCKF)

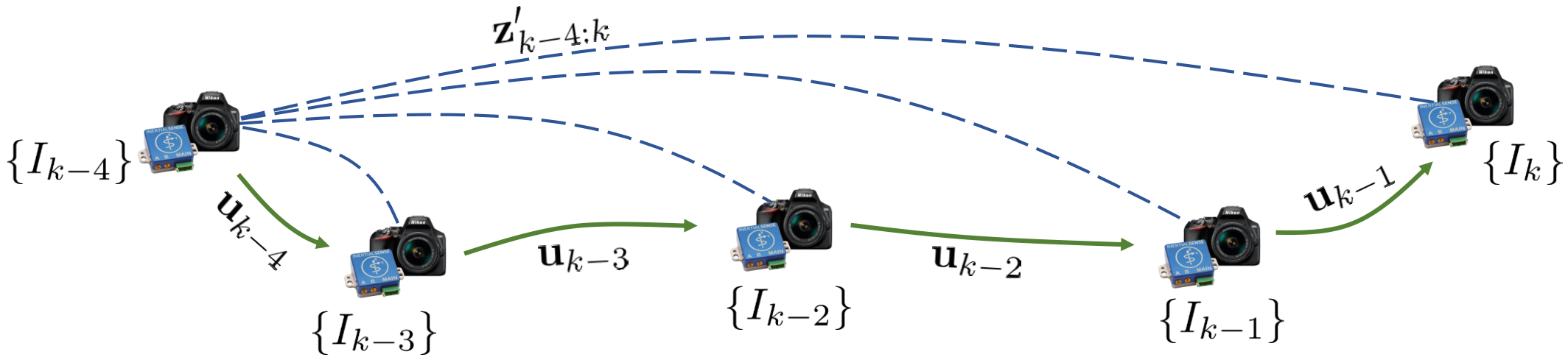
- The MSCKF allows for updating features without inserting their estimates into the state vector $\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_I^\top & \mathbf{x}_C^\top \end{bmatrix}^\top$
- Reduced complexity increases computational efficiency



[1] Mourikis, Anastasios I., and Stergios I. Roumeliotis. "A multi-state constraint Kalman filter for vision-aided inertial navigation." Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 2007.

Multi-State Constraint Kalman Filter (MSCKF)

- The MSCKF allows for updating features without inserting their estimates into the state vector $\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_I^\top & \mathbf{x}_C^\top \end{bmatrix}^\top$
- Reduced complexity increases computational efficiency

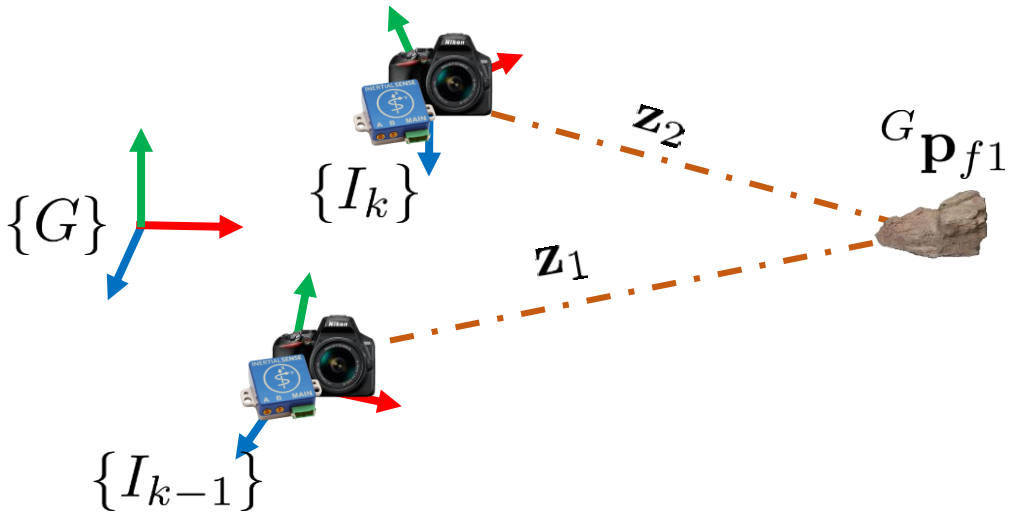


[1] Mourikis, Anastasios I., and Stergios I. Roumeliotis. "A multi-state constraint Kalman filter for vision-aided inertial navigation." Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 2007.

MSCKF Measurement Update

- Measurement equations

$$\mathbf{z}_1 = \mathbf{h}(\mathbf{x}_k, {}^G\mathbf{p}_{f1}) + \mathbf{n}_{pix}$$
$$\mathbf{z}_2 = \mathbf{h}(\mathbf{x}_k, {}^G\mathbf{p}_{f1}) + \mathbf{n}_{pix}$$



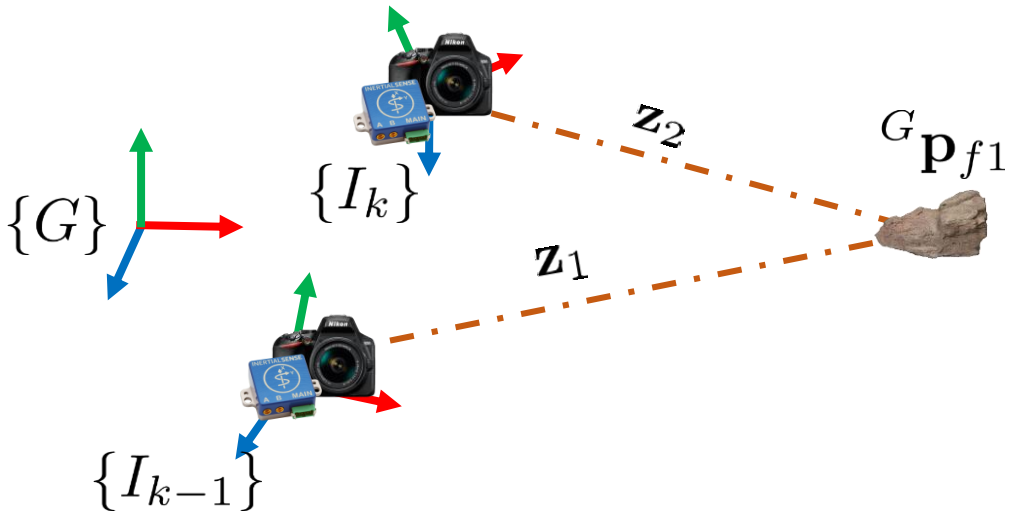
MSCKF Measurement Update

- Measurement equations

$$\mathbf{z}_1 = \mathbf{h}(\mathbf{x}_k, {}^G\mathbf{p}_{f1}) + \mathbf{n}_{pix}$$
$$\mathbf{z}_2 = \mathbf{h}(\mathbf{x}_k, {}^G\mathbf{p}_{f1}) + \mathbf{n}_{pix}$$

- Linearized measurement equation

$$\tilde{\mathbf{z}}_{1:2} = \mathbf{H}_x\tilde{\mathbf{x}}_k + \mathbf{H}_f{}^G\tilde{\mathbf{p}}_{f_1} + \mathbf{n}_{pix}$$



MSCKF Measurement Update

- Measurement equations

$$\mathbf{z}_1 = \mathbf{h}(\mathbf{x}_k, {}^G\mathbf{p}_{f1}) + \mathbf{n}_{pix}$$
$$\mathbf{z}_2 = \mathbf{h}(\mathbf{x}_k, {}^G\mathbf{p}_{f1}) + \mathbf{n}_{pix}$$

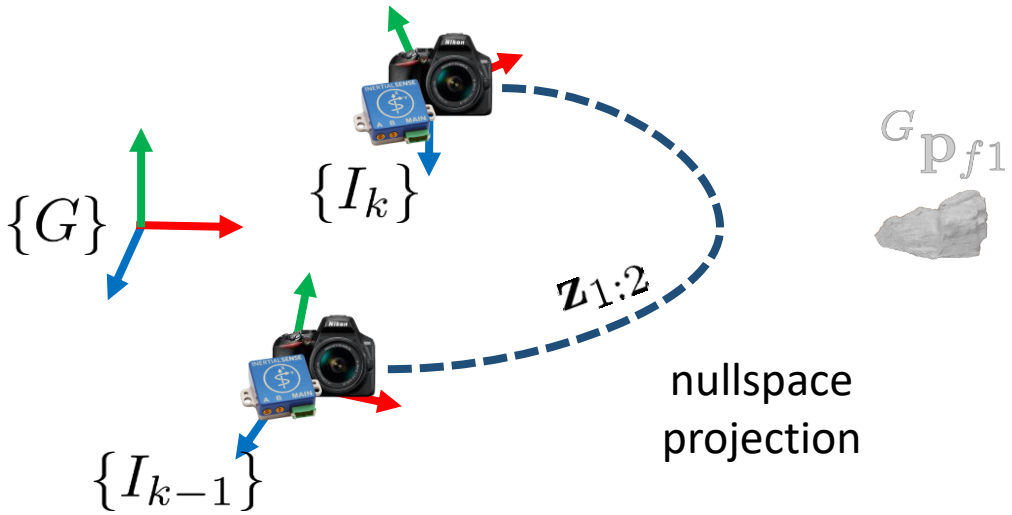
- Linearized measurement equation

$$\tilde{\mathbf{z}}_{1:2} = \mathbf{H}_x\tilde{\mathbf{x}}_k + \mathbf{H}_f{}^G\tilde{\mathbf{p}}_{f_1} + \mathbf{n}_{pix}$$

- Project onto $\mathbf{H}_f$ left nullspace $\mathbf{Q_n}$
  - Equivalent to marginalization

$$\mathbf{Q_n}^\top\tilde{\mathbf{z}}_{1:2} = \mathbf{Q_n}^\top\mathbf{H}_x\tilde{\mathbf{x}}_k + \mathbf{Q_n}^\top\mathbf{H}_f{}^G\tilde{\mathbf{p}}_{f_1} + \mathbf{Q_n}^\top\mathbf{n}_{pix}$$

$$\mathbf{Q_n}^\top\tilde{\mathbf{z}}_{1:2} = \mathbf{Q_n}^\top\mathbf{H}_x\tilde{\mathbf{x}}_k + \mathbf{Q_n}^\top\mathbf{n}_{pix}$$



MSCKF Measurement Update

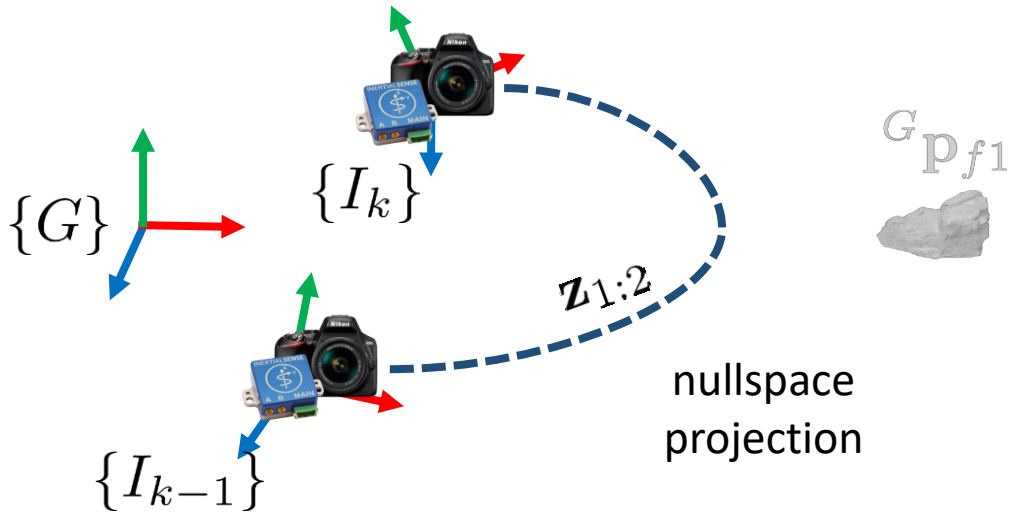
- Measurement equations

$$\mathbf{z}_1 = \mathbf{h}(\mathbf{x}_k, {}^G\mathbf{p}_{f1}) + \mathbf{n}_{pix}$$
$$\mathbf{z}_2 = \mathbf{h}(\mathbf{x}_k, {}^G\mathbf{p}_{f1}) + \mathbf{n}_{pix}$$

- Linearized measurement equation

$$\tilde{\mathbf{z}}_{1:2} = \mathbf{H}_x\tilde{\mathbf{x}}_k + \mathbf{H}_f {}^G\tilde{\mathbf{p}}_{f_1} + \mathbf{n}_{pix}$$

- Project onto $\mathbf{H}_f$ left nullspace $\mathbf{Q_n}$
  - Equivalent to marginalization



$$\mathbf{Q_n}^\top\tilde{\mathbf{z}}_{1:2} = \mathbf{Q_n}^\top\mathbf{H}_x\tilde{\mathbf{x}}_k + \mathbf{Q_n}^\top\mathbf{H}_f {}^G\tilde{\mathbf{p}}_{f_1} + \mathbf{Q_n}^\top\mathbf{n}_{pix}$$

$$\mathbf{Q_n}^\top\tilde{\mathbf{z}}_{1:2} = \mathbf{Q_n}^\top\mathbf{H}_x\tilde{\mathbf{x}}_k + \mathbf{Q_n}^\top\mathbf{n}_{pix}$$

Nullspace projection causes derivative in respect to the feature to go to zero!

MSCKF Measurement Update

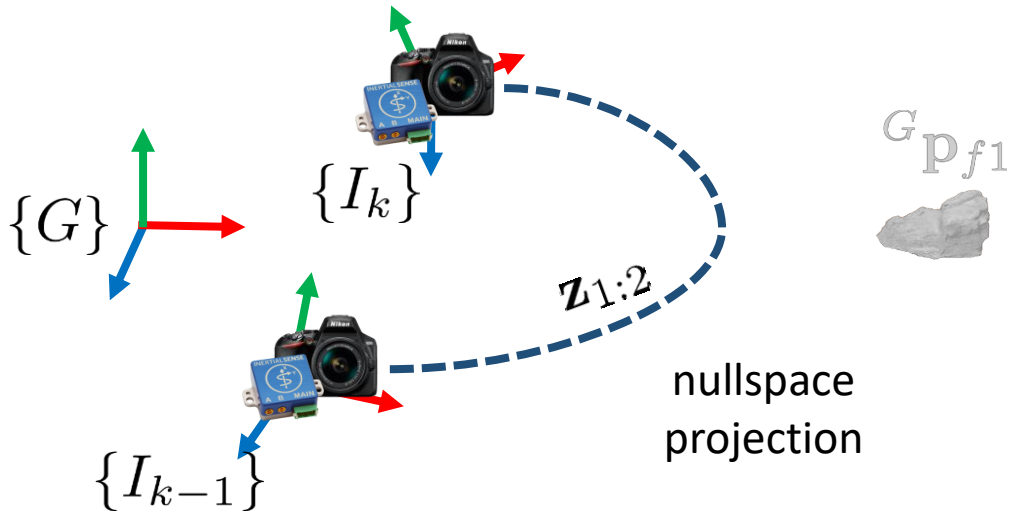
- Measurement equations

$$\mathbf{z}_1 = \mathbf{h}(\mathbf{x}_k, {}^G\mathbf{p}_{f1}) + \mathbf{n}_{pix}$$
$$\mathbf{z}_2 = \mathbf{h}(\mathbf{x}_k, {}^G\mathbf{p}_{f1}) + \mathbf{n}_{pix}$$

- Linearized measurement equation

$$\tilde{\mathbf{z}}_{1:2} = \mathbf{H}_x\tilde{\mathbf{x}}_k + \mathbf{H}_f {}^G\tilde{\mathbf{p}}_{f_1} + \mathbf{n}_{pix}$$

- Project onto $\mathbf{H}_f$ left nullspace $\mathbf{Q_n}$
  - Equivalent to marginalization



$$\mathbf{Q_n}^\top\tilde{\mathbf{z}}_{1:2} = \mathbf{Q_n}^\top\mathbf{H}_x\tilde{\mathbf{x}}_k + \mathbf{Q_n}^\top\mathbf{H}_f {}^G\tilde{\mathbf{p}}_{f_1} + \mathbf{Q_n}^\top\mathbf{n}_{pix}$$

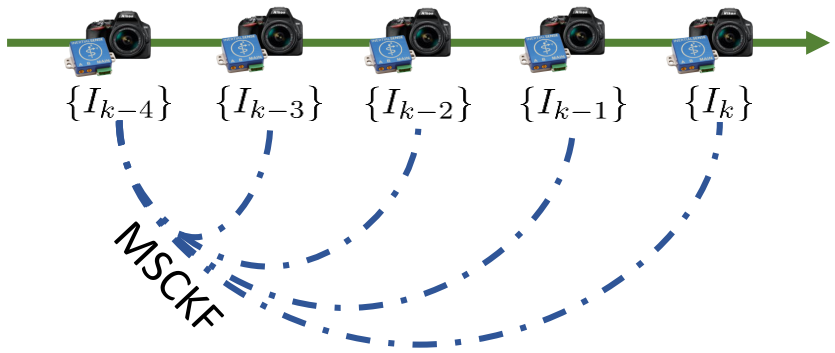
$$\mathbf{Q_n}^\top\tilde{\mathbf{z}}_{1:2} = \mathbf{Q_n}^\top\mathbf{H}_x\tilde{\mathbf{x}}_k + \mathbf{Q_n}^\top\mathbf{n}_{pix}$$

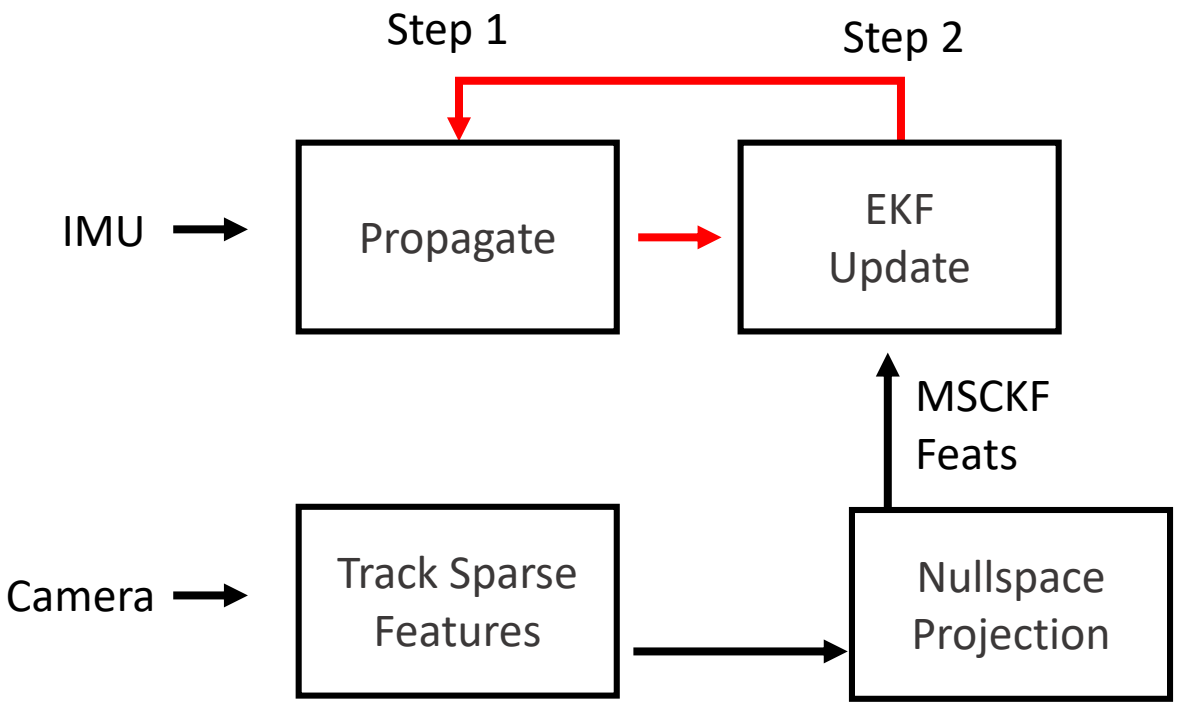
Nullspace projection causes derivative in respect to the feature to go to zero!

- Perform standard EKF update!

MSCKF – Estimation Flowchart: Recap

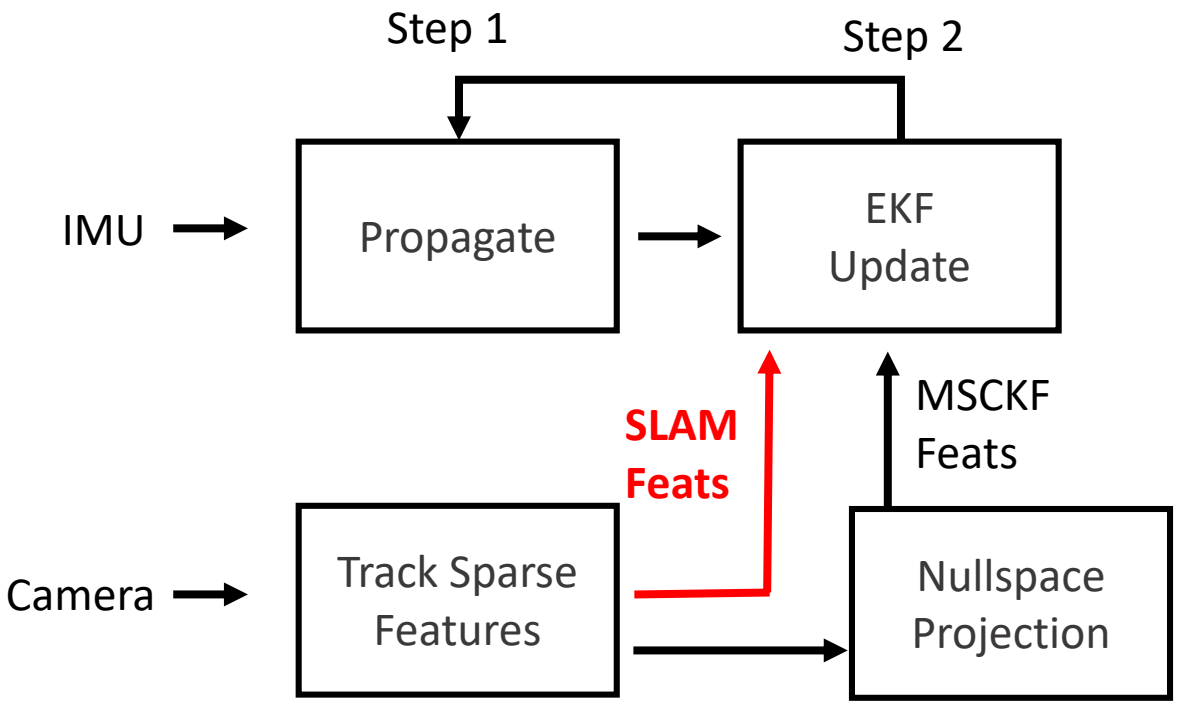
- MSCKF is an efficient method for **light-weight** sensor fusion
- Two step process: Inertial propagation and measurement update

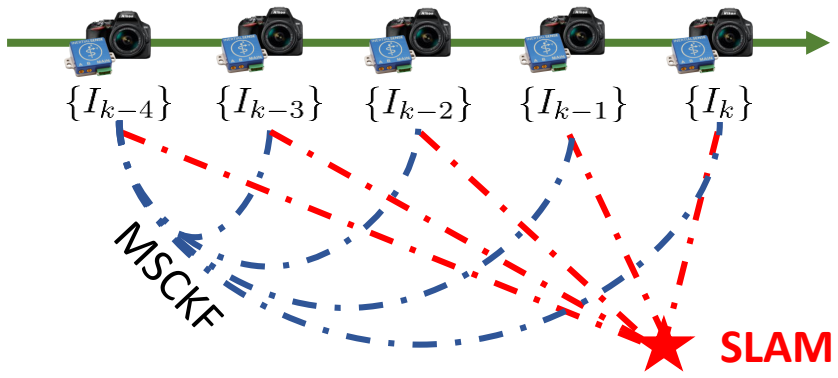




MSCKF – Estimation Flowchart: Recap

- MSCKF is an efficient method for **light-weight** sensor fusion
- Two step process: Inertial propagation and measurement update

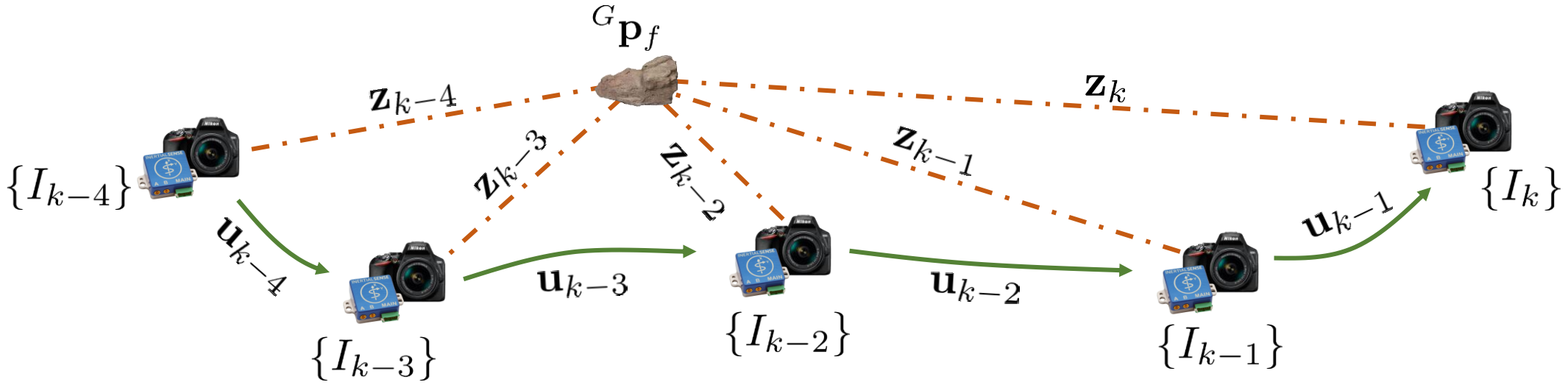




Can also continuously estimate "SLAM" features kept in state

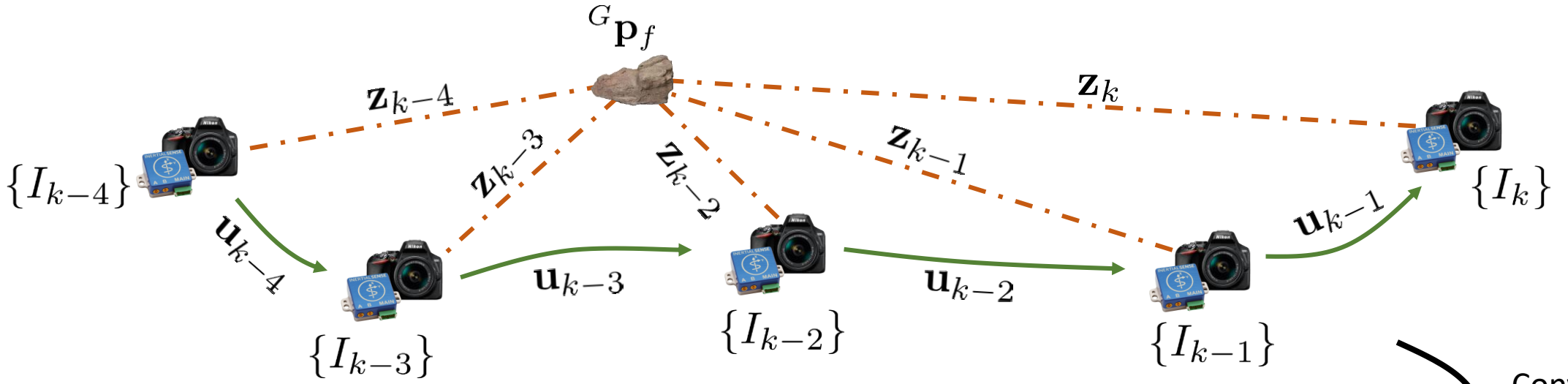
Problem: Factor Graph Perspective

- Robotic Trajectory:



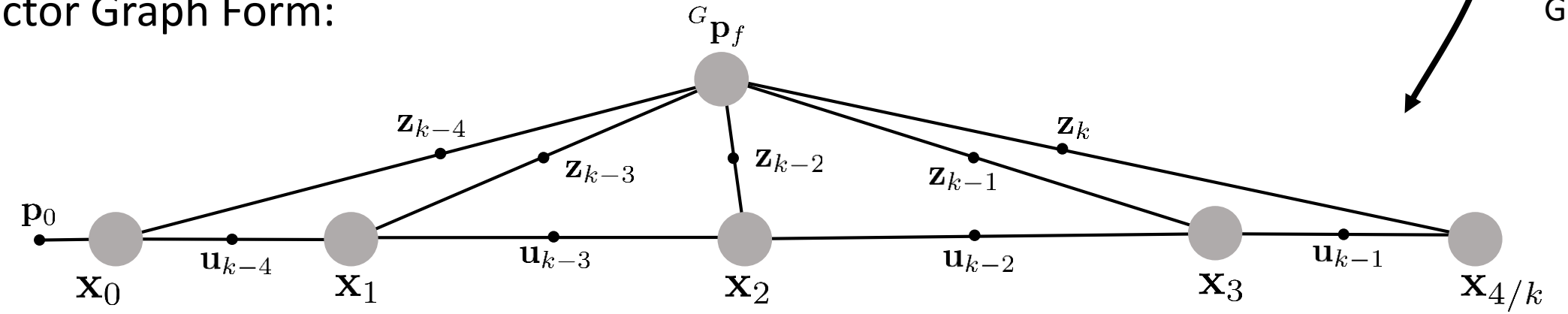
Problem: Factor Graph Perspective

- Robotic Trajectory:



Converted to Factor Graph

- Factor Graph Form:



Batch Least Squares (BLS) Estimation

Prior Cost

$$\mathbf{x}_0 = \mathbf{x}'_0 + \mathbf{n}$$

$$\mathcal{C}_{prior} = ||\mathbf{x}_0 - \mathbf{x}'_0||^2_{\mathbf{P}_0}$$

Inertial Cost

$$\mathbf{x}_k = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{n}_{k-1})$$

control input / state evolution

$$\mathcal{C}_{imu} = ||\mathbf{x}_k - \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{n}_{k-1})||^2_{\mathbf{Q}_k}$$

Feature / Bearing Cost

$$\mathbf{z}_m = \mathbf{h}(\mathbf{x}_k) + \mathbf{n}$$

observation

$$\mathcal{C}_{cam} = ||\mathbf{z}_m - \mathbf{h}(\mathbf{x}_k)||^2_{\mathbf{R}_k}$$

[1] Grisetti, Giorgio, et al. "g2o: A general framework for (hyper) graph optimization." Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China. 2011.
[2] Dellaert, Frank. Factor graphs and GTSAM: A hands-on introduction. Georgia Institute of Technology, 2012.

Batch Least Squares (BLS) Estimation

Prior Cost

$$\mathbf{x}_0 = \mathbf{x}'_0 + \mathbf{n}$$

$$\mathcal{C}_{prior} = ||\mathbf{x}_0 - \mathbf{x}'_0||^2_{\mathbf{P}_0}$$

Inertial Cost

$$\mathbf{x}_k = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{n}_{k-1})$$

control input / state evolution

$$\mathcal{C}_{imu} = ||\mathbf{x}_k - \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{n}_{k-1})||^2_{\mathbf{Q}_k}$$

Feature / Bearing Cost

$$\mathbf{z}_m = \mathbf{h}(\mathbf{x}_k) + \mathbf{n}$$

observation

$$\mathcal{C}_{cam} = ||\mathbf{z}_m - \mathbf{h}(\mathbf{x}_k)||^2_{\mathbf{R}_k}$$

Optimization Formulation:

$$[\mathbf{x}_0^\top, \mathbf{x}_1^\top, ..., \mathbf{x}_k^\top]^\top = \underset{\mathbf{x}_{0..k}}{\mathbf{argmin}} \left(\mathcal{C}_{prior} + \sum_{i=0}^{k} \mathcal{C}_{imu}^{(i)} + \sum_{i=0}^{k} \mathcal{C}_{cam}^{(i)} \right)$$

Can use Gauss-Newton algorithm

[1] Grisetti, Giorgio, et al. "g2o: A general framework for (hyper) graph optimization." Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China. 2011.
[2] Dellaert, Frank. Factor graphs and GTSAM: A hands-on introduction. Georgia Institute of Technology, 2012.

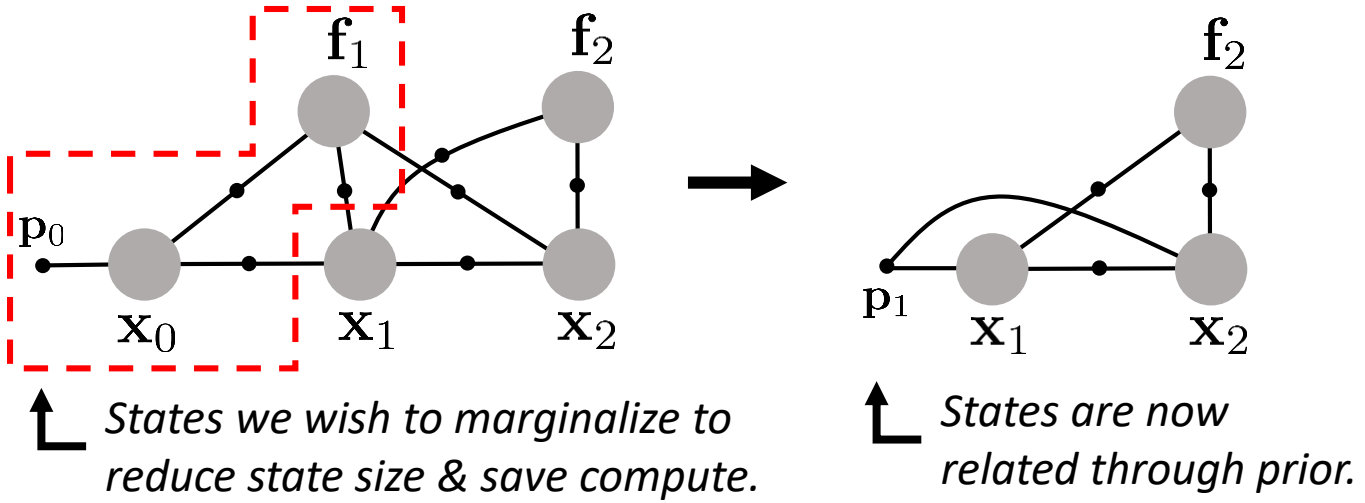
BLS – Efficient Estimators

Challenge:

- Standard BLS is cubic in terms of the state size **$O(n^3)$**
- Application to ***real-time*** robotic systems requires modifications

Solutions:

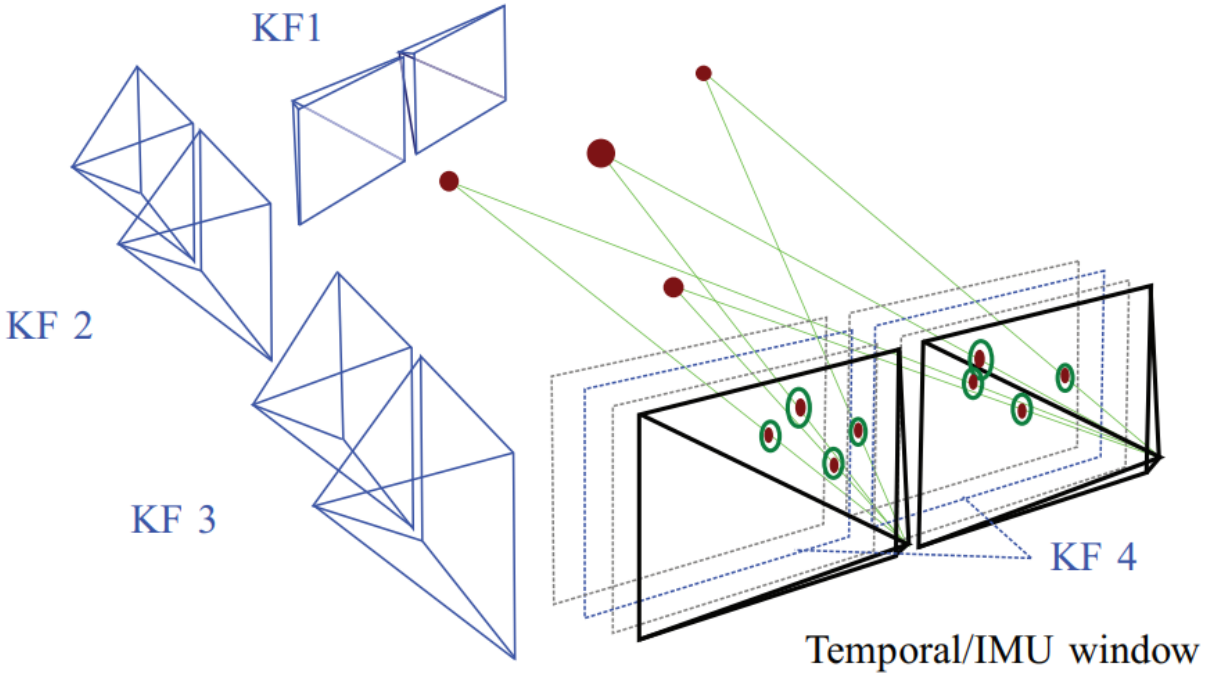
- Sliding window optimization (fixed-lag smoothers)
- BLS with graph reduction (sparsification)
- Incremental optimization



[1] Leutenegger, Stefan, et al. "Keyframe-based visual–inertial odometry using nonlinear optimization." The International Journal of Robotics Research 34.3 (2015): 314-334.
[2] Kaess, Michael, et al. "iSAM2: Incremental smoothing and mapping using the Bayes tree." The International Journal of Robotics Research 31.2 (2012): 216-235.
[3] Hsiung, Jerry, et al. "Information sparsification in visual-inertial odometry." 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.

BLS – Marginalization

- Many different practical methods to enable efficient BLS
- Key Considerations:
  - What states to marginalize?
  - When to marginalize?
- Examples:
  - Keyframing of poses
  - Dropping of features
  - Duplication of measurements
  - Non-linear factor recovery
  - Fully dense marginal factor

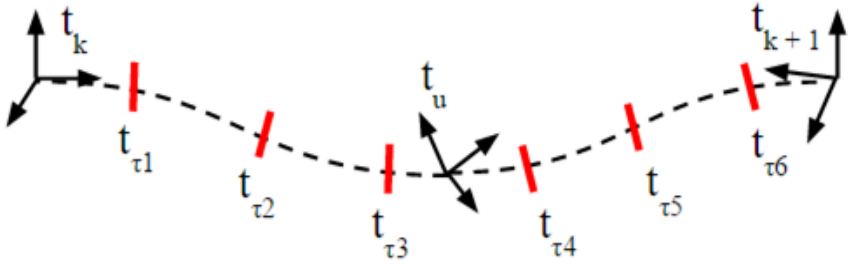


[photo] Leutenegger, Stefan, et al. "Keyframe-based visual–inertial odometry using nonlinear optimization." The International Journal of Robotics Research 34.3 (2015): 314-334.
[2] Nerurkar, Esha D., Kejian J. Wu, and Stergios I. Roumeliotis. "C-KLAM: Constrained keyframe-based localization and mapping." 2014 IEEE international conference on robotics and automation (ICRA). IEEE, 2014.

BLS – Inertial Pre-integration

- Naive use of inertial measurement model requires **re-integration** of $\mathcal{C}_{imu}$
- Decouple integration and state variables (through approximations) to ***remove*** need to re-integrate when states re-linearize during optimization

$$\mathcal{C}_{imu} = ||\mathbf{x}_k - \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{n}_{k-1})||^2_{\mathbf{Q}_k}$$



$${}^{k+1}_{G}\mathbf{R} = {}^{k+1}_{k}\mathbf{R}\,{}^{k}_{G}\mathbf{R}$$

Not a function of current state (after bias linear approx)!

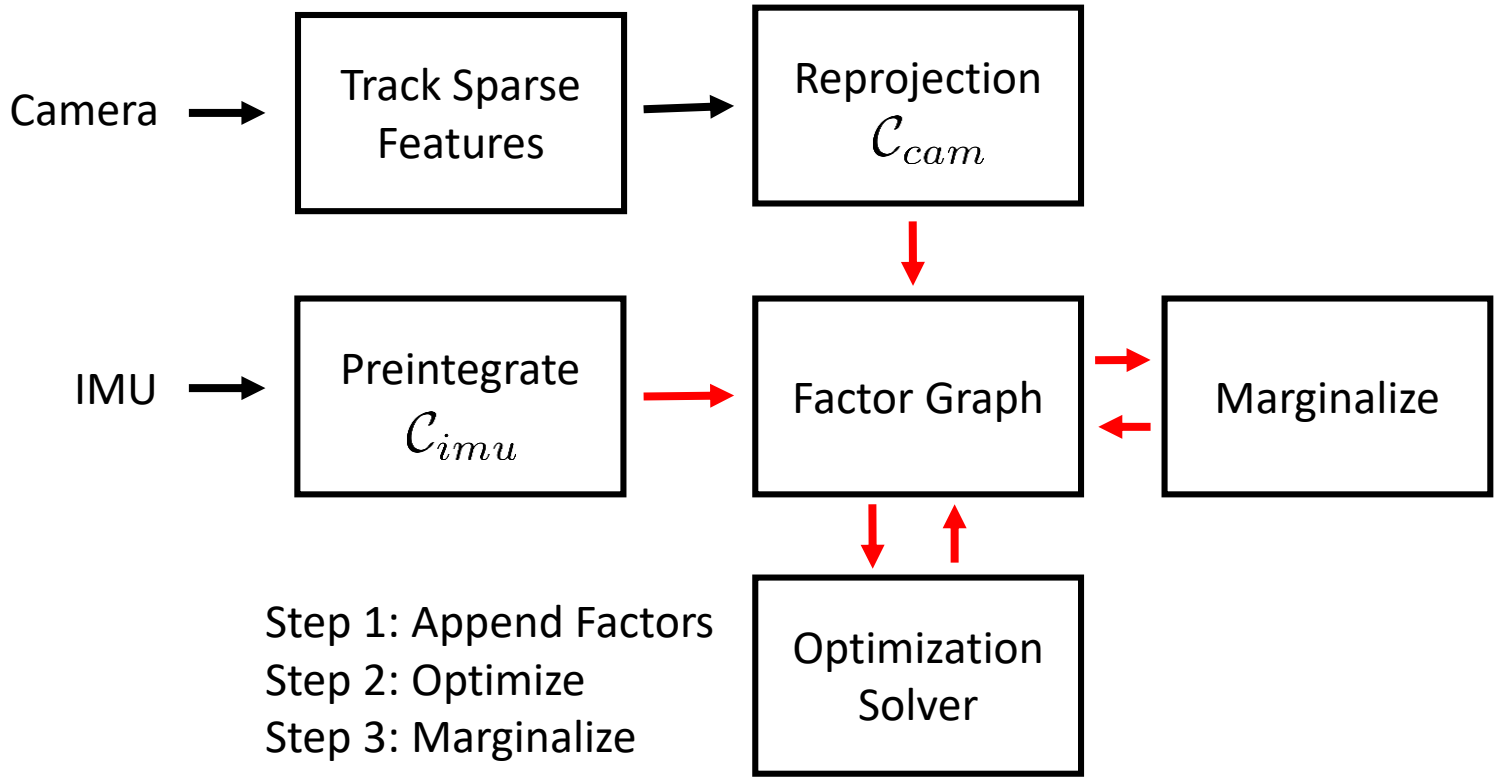
$${}^{G}\mathbf{v}_{k+1} = {}^{G}\mathbf{v}_k - {}^{G}\mathbf{g}\Delta T + {}^{G}_{k}\mathbf{R}\underbrace{\int_{t_k}^{t_{k+1}} {}^{k}_{u}\mathbf{R}\,({}^{u}\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a)\,du}_{{}^{k}\boldsymbol{\beta}_{k+1}}$$

$${}^{G}\mathbf{p}_{k+1} = {}^{G}\mathbf{p}_k + {}^{G}\mathbf{v}_k\Delta T - \frac{1}{2}{}^{G}\mathbf{g}\Delta T^2 + {}^{G}_{k}\mathbf{R}\underbrace{\int_{t_k}^{t_{k+1}}\int_{t_k}^{s} {}^{k}_{u}\mathbf{R}\,({}^{u}\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a)\,duds}_{{}^{k}\boldsymbol{\alpha}_{k+1}}$$

[1] Forster, Christian, et al. "On-manifold preintegration theory for fast and accurate visual-inertial navigation." IEEE Transactions on Robotics (2015): 1-18.
[2] Eckenhoff, Kevin, Patrick Geneva, and Guoquan Huang. "Closed-form preintegration methods for graph-based visual–inertial navigation." The International Journal of Robotics Research 38.5 (2019): 563-586.

BLS – Estimation Flowchart: Recap

- Factor graphs can provide a **unified design language** for sensor fusion algorithms
- Allows for higher level abstraction away from linear algebra representation



Incremental Optimization using Sqrt-Info

- Key ideas:
  - Use Cholesky factorization of Hessian (information)
  - Can formulate **equivalent** optimization problem using square-root matrix
- Advantages of square-root form:
  - Better system **numerical properties** (smaller condition number) can enable use of **single-precision** arithmetic
  - Optimal state ordering can allow for efficient Givens QR (e.g. iSAM)

Standard BLS solution

$$\underset{\mathbf{x}}{\operatorname{argmin}} ||\mathbf{H}(\mathbf{x} - \hat{\mathbf{x}}) - \mathbf{r}||^2 \quad (1)$$

$$\mathbf{H}^\top \mathbf{H} \Delta \mathbf{x} = \mathbf{H}^\top \mathbf{r}$$

Square-root information through QR decomposition

$$\mathbf{H} = \mathbf{Q}\mathbf{U}$$

$$\mathbf{U}^\top \mathbf{U} \Delta \mathbf{x} = \mathbf{U}^\top \mathbf{Q}^\top \mathbf{r}$$

$$\mathbf{U} \Delta \mathbf{x} = \mathbf{Q}^\top \mathbf{r}$$

Reformulate original BLS in square-root form (equivalent)

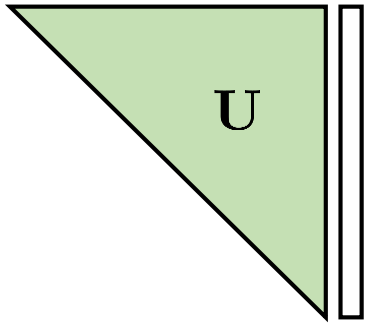
$$\underset{\mathbf{x}}{\operatorname{argmin}} ||\mathbf{U}(\mathbf{x} - \hat{\mathbf{x}}) - \mathbf{Q}^\top \mathbf{r}||^2 \quad (2)$$

[1] Dellaert, Frank, and Michael Kaess. "Square Root SAM: Simultaneous localization and mapping via square root information smoothing." The International Journal of Robotics Research 25.12 (2006): 1181-1203.
[2] Maybeck, Peter S. Stochastic models, estimation, and control, vol 1. Academic press, 1982.
[3] Kaess, Michael, Ananth Ranganathan, and Frank Dellaert. "iSAM: Incremental smoothing and mapping." IEEE Transactions on Robotics 24.6 (2008): 1365-1378.
[4] Wu, Kejian, et al. "A Square Root Inverse Filter for Efficient Vision-aided Inertial Navigation on Mobile Devices." Robotics: Science and Systems. Vol. 2. 2015.

Incremental Optimization – An Example

Prior Cost

$$\mathbf{x} = \hat{\mathbf{x}} + \mathbf{n} \quad \rightarrow \quad \mathcal{C}_{prior} = ||\mathbf{U}(\mathbf{x} - \hat{\mathbf{x}})||^2$$



- All information stored in sqrt information form $\mathbf{P}^{-1} = \mathbf{U}^\top \mathbf{U}$

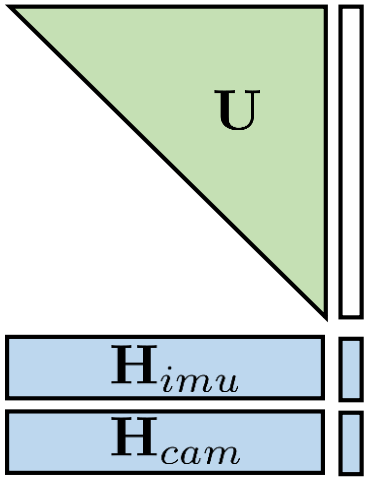
[1] Dellaert, Frank, and Michael Kaess. "Square Root SAM: Simultaneous localization and mapping via square root information smoothing." The International Journal of Robotics Research 25.12 (2006): 1181-1203.
[2] Wu, Kejian, et al. "A Square Root Inverse Filter for Efficient Vision-aided Inertial Navigation on Mobile Devices." Robotics: Science and Systems. Vol. 2. 2015.
[3] Golub, Gene H., and Charles F. Van Loan. Matrix computations. Vol. 3. JHU press, 2013.

Incremental Optimization – An Example

Prior Cost

$$\mathbf{x} = \hat{\mathbf{x}} + \mathbf{n} \quad \rightarrow \quad \mathcal{C}_{prior} = ||\mathbf{U}(\mathbf{x} - \hat{\mathbf{x}})||^2$$

- All information stored in sqrt information form $\mathbf{P}^{-1} = \mathbf{U}^\top \mathbf{U}$



Inertial & Feature Cost

$$\mathbf{x}_k = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{n}_{k-1})$$
$$\mathbf{z}_m = \mathbf{h}(\mathbf{x}_k) + \mathbf{n}$$
$$\rightarrow \quad \mathcal{C}_{imu/cam} = ||\mathbf{H}(\mathbf{x} - \hat{\mathbf{x}}) - \mathbf{r}||^2$$

New Factors:

- Will ruin upper triangle structure
- Re-triangulate using QR
- Appends new information to sqrt matrix!
- State re-ordering allows for efficient QR

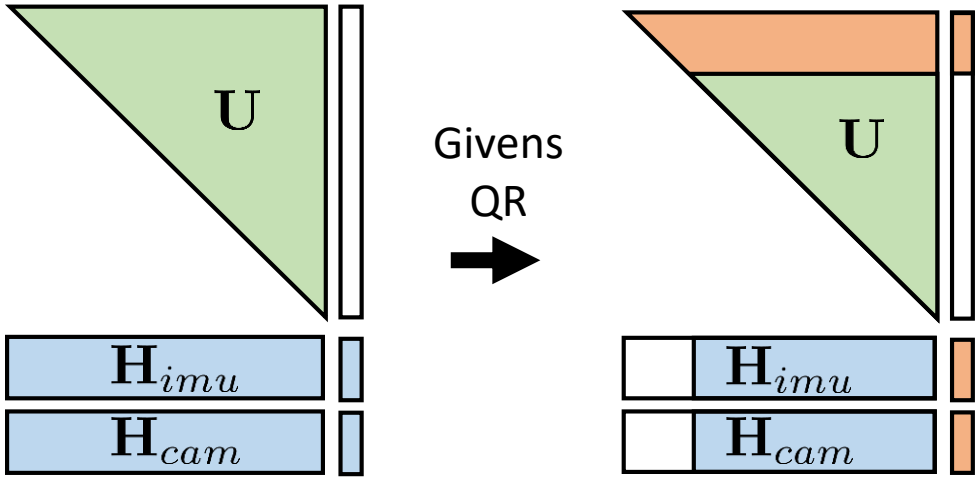
[1] Dellaert, Frank, and Michael Kaess. "Square Root SAM: Simultaneous localization and mapping via square root information smoothing." The International Journal of Robotics Research 25.12 (2006): 1181-1203.
[2] Wu, Kejian, et al. "A Square Root Inverse Filter for Efficient Vision-aided Inertial Navigation on Mobile Devices." Robotics: Science and Systems. Vol. 2. 2015.
[3] Golub, Gene H., and Charles F. Van Loan. Matrix computations. Vol. 3. JHU press, 2013.

Incremental Optimization – An Example

Prior Cost

$$\mathbf{x} = \hat{\mathbf{x}} + \mathbf{n} \quad \rightarrow \quad \mathcal{C}_{prior} = ||\mathbf{U}(\mathbf{x} - \hat{\mathbf{x}})||^2$$

- All information stored in sqrt information form $\mathbf{P}^{-1} = \mathbf{U}^\top \mathbf{U}$



Inertial & Feature Cost

$$\mathbf{x}_k = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{n}_{k-1})$$
$$\mathbf{z}_m = \mathbf{h}(\mathbf{x}_k) + \mathbf{n}$$
$$\rightarrow \quad \mathcal{C}_{imu/cam} = ||\mathbf{H}(\mathbf{x} - \hat{\mathbf{x}}) - \mathbf{r}||^2$$

New Factors:

- Will ruin upper triangle structure
- Re-triangulate using QR
- Appends new information to sqrt matrix!
- State re-ordering allows for efficient QR

[1] Dellaert, Frank, and Michael Kaess. "Square Root SAM: Simultaneous localization and mapping via square root information smoothing." The International Journal of Robotics Research 25.12 (2006): 1181-1203.
[2] Wu, Kejian, et al. "A Square Root Inverse Filter for Efficient Vision-aided Inertial Navigation on Mobile Devices." Robotics: Science and Systems. Vol. 2. 2015.
[3] Golub, Gene H., and Charles F. Van Loan. Matrix computations. Vol. 3. JHU press, 2013.

Estimation Methodology Equivalences

EKF ⇔ MAP Optimization w/ one Gauss-Newton Iteration

EKF ⇔ Extended Inverse (Information) Filter (EIF)

EKF ⇔ Square-Root EKF (SW-EKF)

MSCKF (nullspace) ⇔ BLS with Feature Marginalization (schur)

Takeaway:

Equivalent up to linearization errors in theory. Choice is based on end application use (e.g., computational efficiency and accuracy levels required).

[1] Bell, Bradley M., and Frederick W. Cathey. "The iterated Kalman filter update as a Gauss-Newton method." IEEE Transactions on Automatic Control 38.2 (1993): 294-297.
[2] Wu, Kejian, et al. "A Square Root Inverse Filter for Efficient Vision-aided Inertial Navigation on Mobile Devices." Robotics: Science and Systems. Vol. 2. 2015.
[3] Yang, Yulin, James Maley, and Guoquan Huang. "Null-space-based marginalization: Analysis and algorithm." 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017.

Outline – Essential Building Blocks

- Feature Tracking and Matching
- Observability
- Filter Consistency
- Degenerate Motion
- Initialization
- Calibration
- Robustness
- Long-Term Navigation

Feature Tracking and Matching

- Indirect methods
  - Geometric points or lines
  - Track temporally using KLT or descriptors
  - RANSAC for outlier rejection
- Direct methods
  - Intensity based cost function
  - Still rely on gradient information, but more robust in low-texture environments
  - How to project features into future frames?

[1] Lucas, Bruce D., and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." 1981.
[2] Rublee, Ethan, et al. "ORB: An efficient alternative to SIFT or SURF." 2011 International conference on computer vision. Ieee, 2011.
[3] Golub, Gene H., and Charles F. Van Loan. Matrix computations. Vol. 3. JHU press, 2013.
[4] Bloesch, Michael, et al. "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback." The International Journal of Robotics Research 36.10 (2017): 1053-1072.

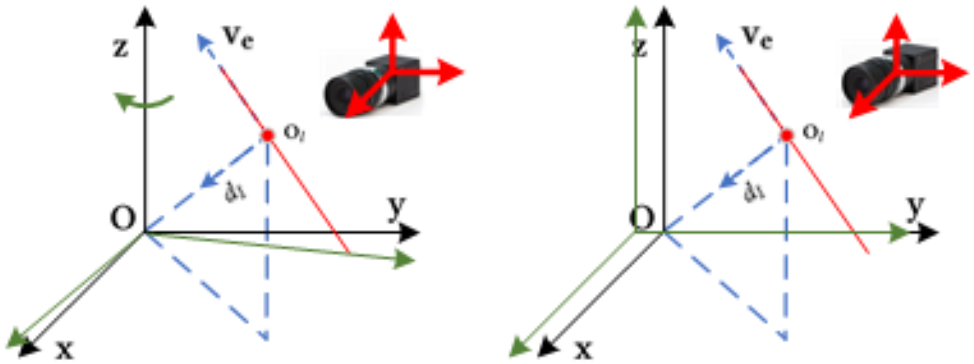
Observability

- Determine if we are able to **fully recover** the state given sensor **measurements**

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 \\ \mathbf{M}_2 \\ \vdots \\ \mathbf{M}_k \end{bmatrix} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \mathbf{\Phi}_{(2,1)} \\ \vdots \\ \mathbf{H}_k \mathbf{\Phi}_{(k,1)} \end{bmatrix}$$

  - Compute by stacking all Jacobians and state transitions
  - If nullspace exists then it is the unobs. direction $\mathbf{MN} \stackrel{?}{=} \mathbf{0}$

- Why we care about observability:
  - Determines **minimal** information to recover states
  - Enables design of **consistent** estimators (e.g., FEJ, OC-EKF, etc.)
  - Identify **degenerate** motions



Conical 4 unobs. directions for VINS:
Yaw (left), Translation (right)

[1] Yang, Yulin, and Guoquan Huang. "Observability analysis of aided ins with heterogeneous features of points, lines, and planes." IEEE Transactions on Robotics 35.6 (2019): 1399-1418.

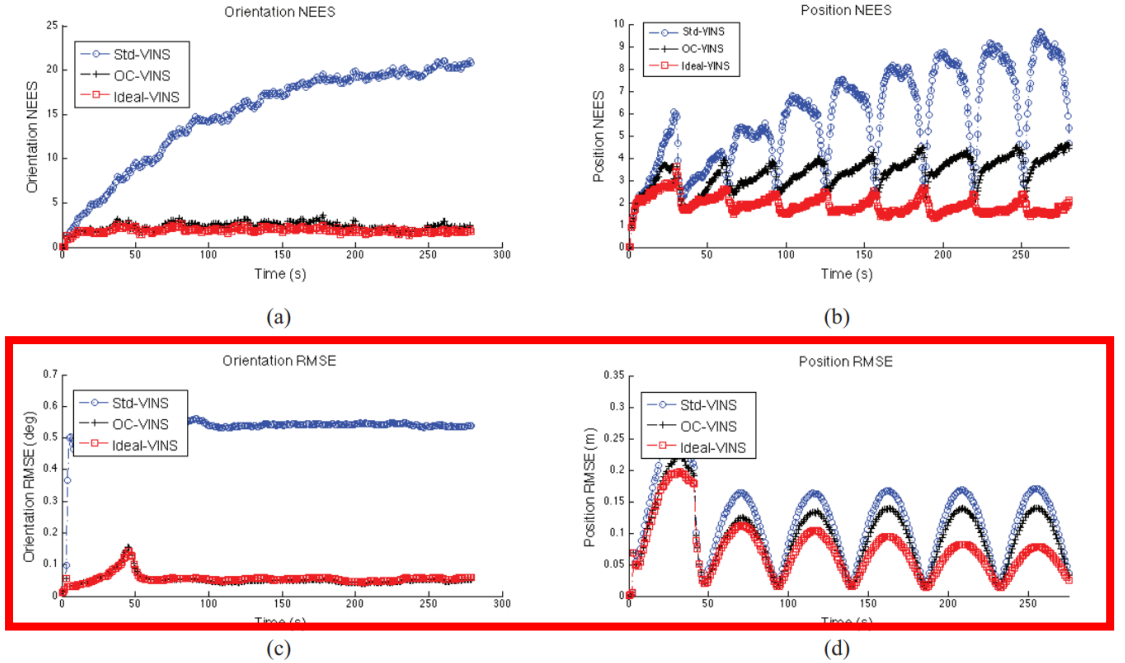
Estimator Consistency

- Estimation error should be **zero mean** and estimate **covariance** should be **larger than or equal** to the true covariance
- Preventing information gain in unobservable directions is key to improving consistency
- Existing Algorithms:
  - Robot-centric
  - First-Estimate Jacobians
  - OC-EKF
  - Invariant filters

All preserve original VINS unobs. dir.

$$e_{nees} = (\mathbf{x} - \hat{\mathbf{x}})^\top \mathbf{P}^{-1} (\mathbf{x} - \hat{\mathbf{x}})$$

NEES is large since covariance is overconfident



Inconsistency cause estimator errors, thus we minimize inconsistencies!

[1] Castellanos, José A., José Neira, and Juan D. Tardós. "Limits to the consistency of EKF-based SLAM." IFAC Proceedings Volumes 37.8 (2004): 716-721.
[2] Huang, Guoquan P., Anastasios I. Mourikis, and Stergios I. Roumeliotis. "Observability-based rules for designing consistent EKF SLAM estimators." The International Journal of Robotics Research 29.5 (2010): 502-528.
[3] Wu, Kanzhi, et al. "An invariant-EKF VINS algorithm for improving consistency." 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017.
[4] Bar-Shalom, Yaakov, X. Rong Li, and Thiagalingam Kirubarajan. Estimation with applications to tracking and navigation: theory algorithms and software. John Wiley & Sons, 2004.
[photo] Hesch, Joel A., et al. "Camera-IMU-based localization: Observability analysis and consistency improvement." The International Journal of Robotics Research 33.1 (2014): 182-201.

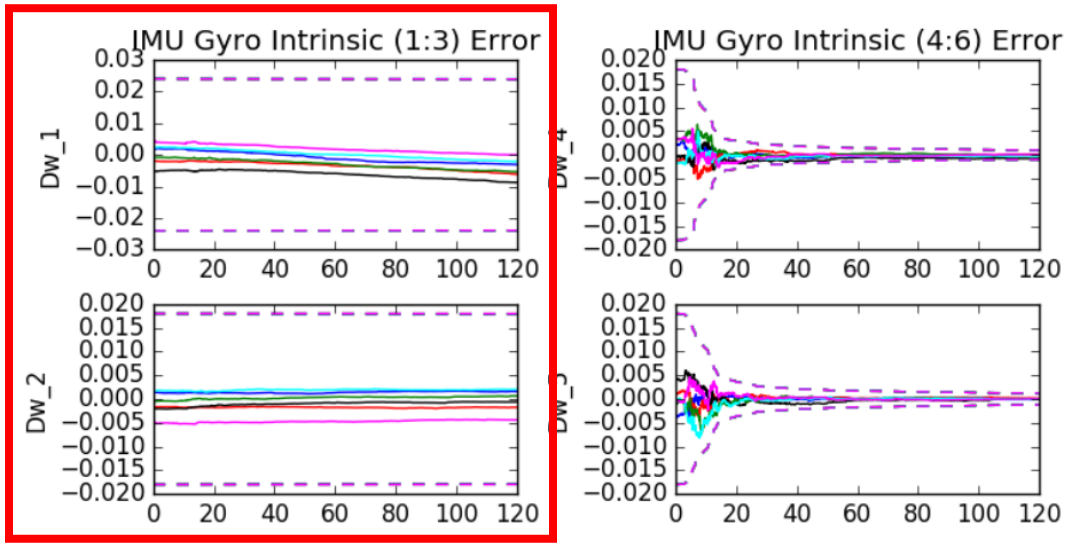
Degeneracy

- Situations which cause **additional unobservable** directions in VINS (4DoF canonical)
- Identify degeneracies through inspection of observability matrix
  - System – Likely to cause VINS to fail
  - Calibration – Can possibly degrade performance
- Degeneracies weaken system robustness
- Degeneracies can **test estimator consistency** since there should be no information gained

System Degeneracies

Motion	Sensor	Unobservable
1. Pure translation	General	Orientation ${}^{I}_{G}\mathbf{R}$
2. Constant acceleration	Mono cam	System scale
3. Pure rotation	Mono cam	Feature scale
4. Toward the feature	Mono cam	Feature scale

Calibration Degeneracies



No information is gained in unobservable directions (constant variance)

[1] Yang, Yulin, et al. "Degenerate motion analysis for aided ins with online spatial and temporal sensor calibration." IEEE Robotics and Automation Letters 4.2 (2019): 2070-2077.
[2] Yang, Yulin, et al. "Online IMU Intrinsic Calibration: Is It Necessary?." Proc. of the Robotics: Science and Systems, Corvallis, Oregon (2020): 716-20.

State Initialization

- Initialization is the task of determining the initial system state
- VINS has 4DoF unobservable, thus need to initialize the other 11DoF
- Initialization Challenges:
  - Want to initialize as **fast** and **robustly** as possible
  - Shorter time makes recovering the initial states more difficult or unobs.
  - Longer times introduce error due to time offsets, inertial noise, along with increased computation

$$\mathbf{x}_{I_0} = \begin{bmatrix} {}^{I_0}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_0}^\top & {}^{G}\mathbf{v}_{I_0}^\top & \mathbf{b}_{\omega_0}^\top & \mathbf{b}_{a_0}^\top \end{bmatrix}^\top$$

Global yaw and position unobservable (thus can be chosen arbitrarily)

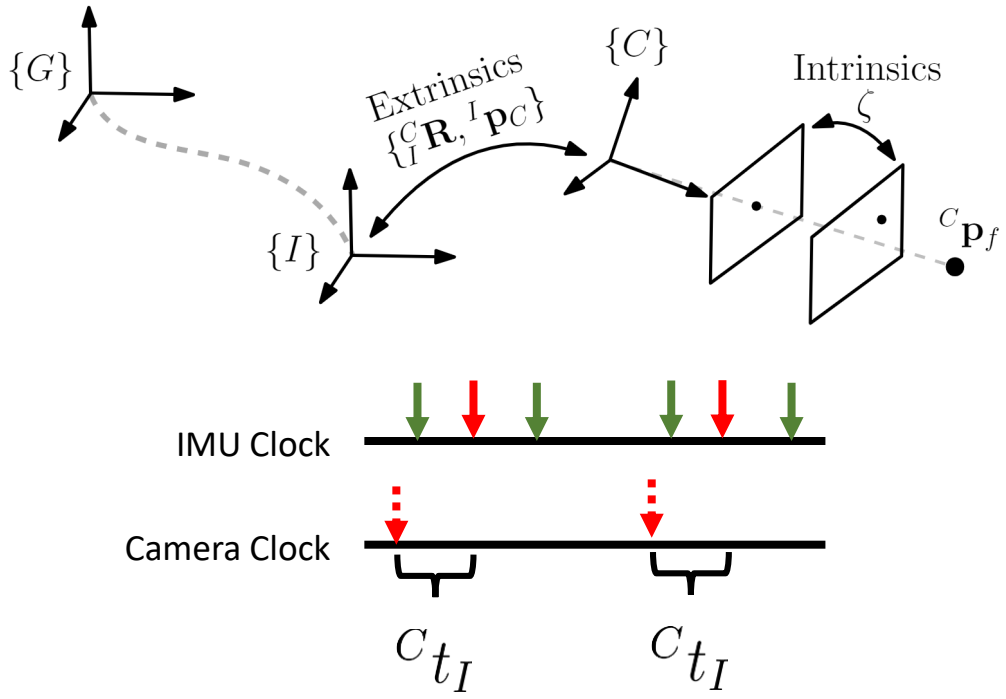
Example SFM Procedure:

1. Collect a window of measurements
2. Perform traditional visual SFM to get up-to-scale camera trajectory
3. With pre-integrated inertial readings recover gravity and state velocities
4. Refine estimates with non-linear optimization to get final state estimates

[1] Martinelli, Agostino. "Closed-form solution of visual-inertial structure from motion." International journal of computer vision 106.2 (2014): 138-152.
[2] Dong-Si, Tue-Cuong, and Anastasios I. Mourikis. "Estimator initialization in vision-aided inertial navigation with unknown camera-IMU calibration." 2012 IROS. IEEE, 2012.
[3] Campos, Carlos, José MM Montiel, and Juan D. Tardós. "Inertial-Only Optimization for Visual-Inertial Initialization." 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020.

Calibration

- Can be performed **offline** prior to estimation, **online** during, or both
- Offline calibration:
  - Highly ***accurate***
  - Can control sensor motion
  - Might not always be possible
- Online calibration:
  - Crucial for ***practical deployments*** handling of poor initial values
  - Handling time-varying calibration parameters
  - Improves estimation robustness

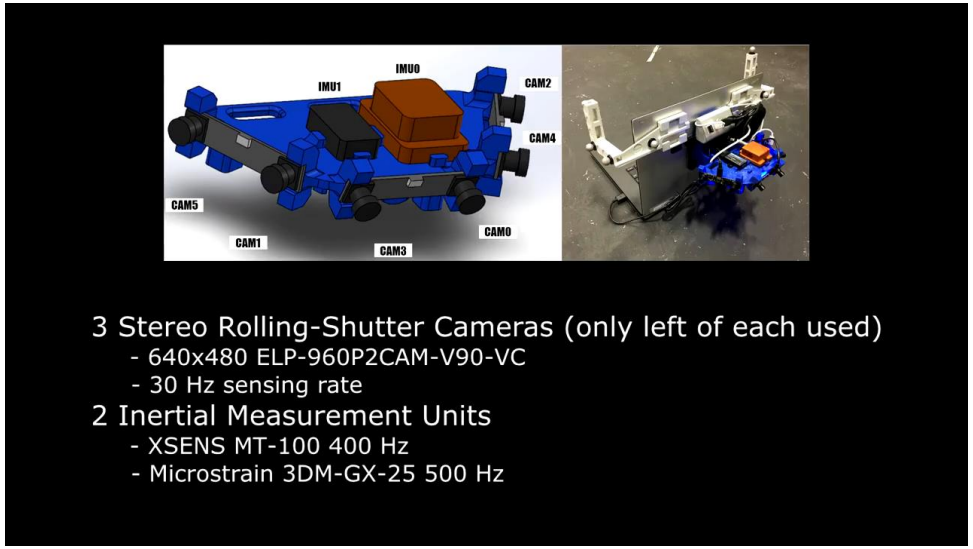


Example Parameters

Camera-inertial spatial-temporal
Camera intrinsic (focal, center, dist)
Inertial intrinsic (scale, skew)

Robustness and Resiliency

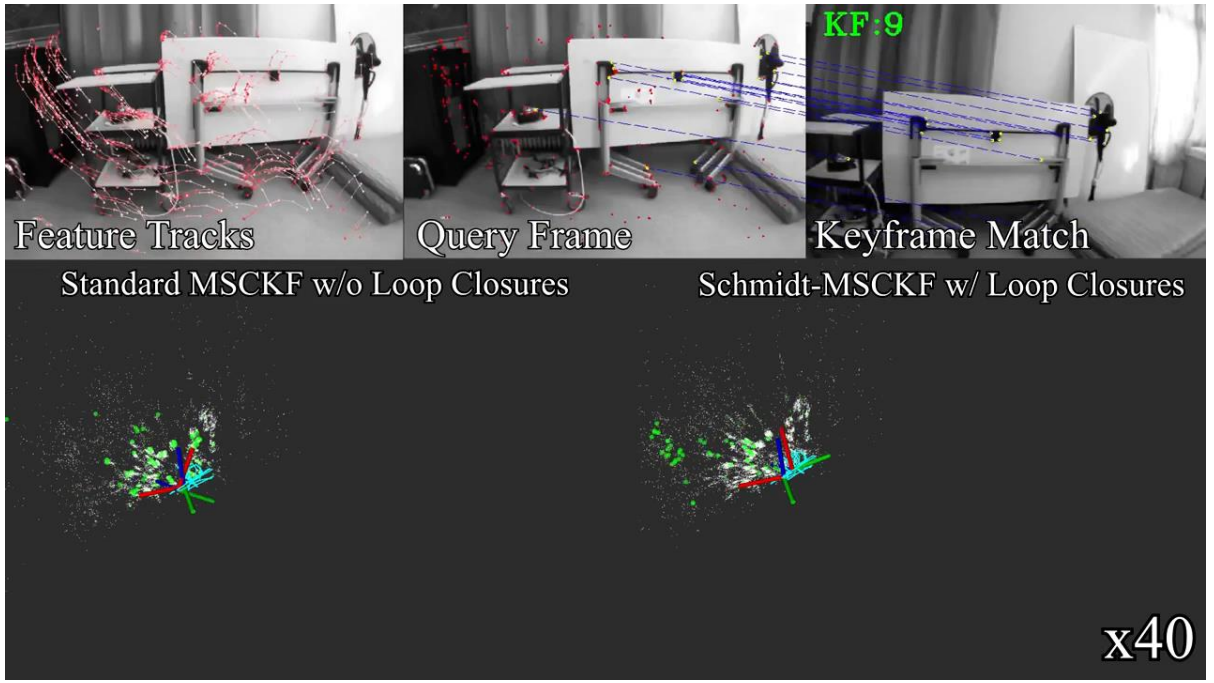
- Challenges:
  1. Hard failures – no measurement information (**sensor drop**)
  2. Soft failures – data becomes **corrupt** (invalidated measurement model)
- Examples:
  - Unmeasurable external forces (e.g., moving platform)
  - Dynamic environments
  - Sensor variations (e.g., exposure, temperature)
- Can address through leveraging multi-sensor fusion



[1] Eckenhoff, Kevin, Patrick Geneva, and Guoquan Huang. "MIMC-VINS: A versatile and resilient multi-IMU multi-camera visual-inertial navigation system." IEEE Transactions on Robotics (2021).

Long-Term Navigation

- Can incorporate loop-closure directly or split the problem into two parts:
  - Frontend (**Localization**): Fast, drifts with time, short-term accuracy matters
  - Backend (**Mapping**): Slow, loop-closure, global consistency matters
- Challenges:
  - Incremental vs full batch pose graph
  - Optimal selection problem to reduce **complexity** and **memory** usage
  - Robust **loop-closure detection** and constraints



Addition of loop-closures can limit odometry drift

[1] Qin, Tong, Peiliang Li, and Shaojie Shen. "Vins-mono: A robust and versatile monocular visual-inertial state estimator." IEEE Transactions on Robotics 34.4 (2018): 1004-1020.
[2] Geneva, Patrick, Kevin Eckenhoff, and Guoquan Huang. "A linear-complexity EKF for visual-inertial navigation with loop closures." 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019.

Outline - Available Open Sourced Systems

- Open Source Estimators
- Dataset Benchmarks
- Metrics and Evaluation

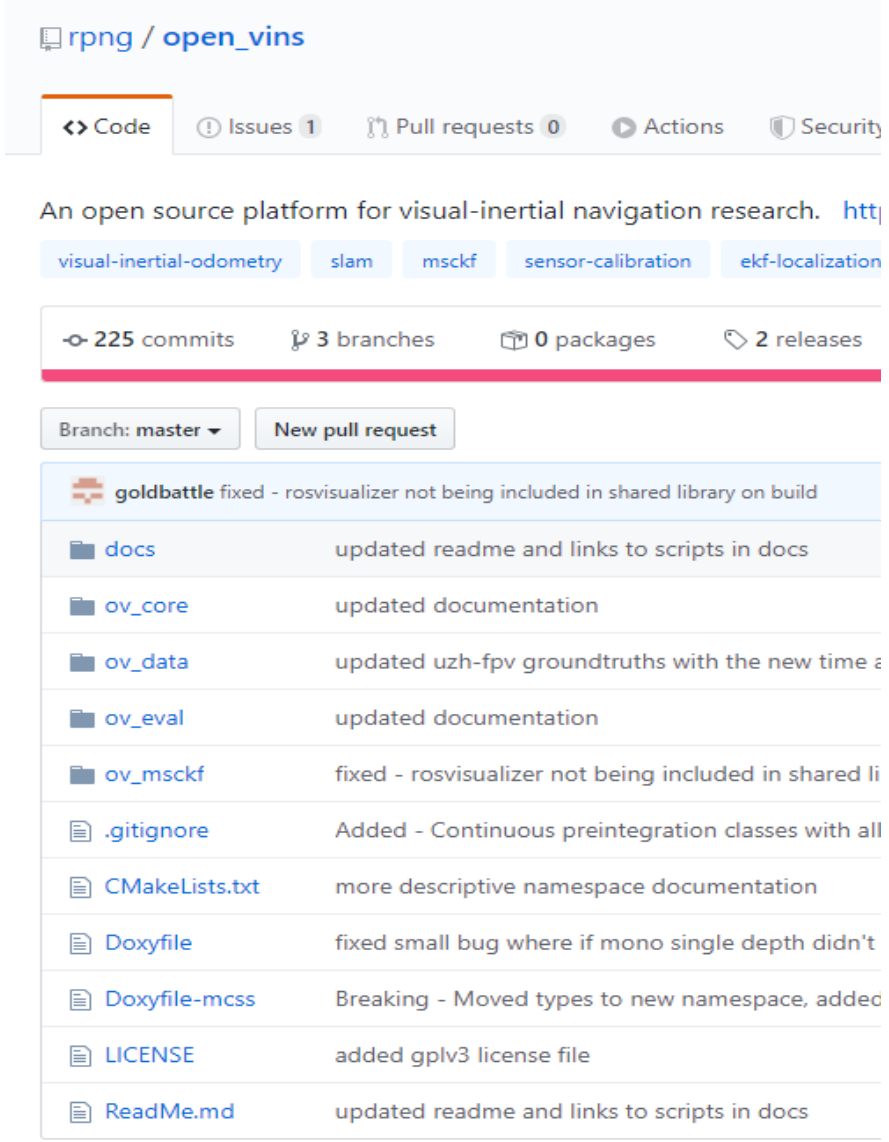
Visual-Inertial Research: Embracing Open Source

System	Mono?	Stereo?	EKF?	Indirect?	Calib. Spatial?	Calib. Intrinsics?	Calib. Time?
OpenVINS	✓	✓	✓	✓	✓	✓	✓
S-MSCKF	-	✓	✓	✓	✓	-	-
R-VIO	✓	-	✓	✓	-	-	-
Rovioli	✓	-	✓	-	✓	-	-
XIVO	✓	-	✓	✓	✓	✓	✓
VINS-Fusion	✓	✓	-	✓	✓	-	✓
OKVIS	✓	✓	-	✓	✓	-	-
Basalt	-	✓	-	✓	-	-	-
ICE-BA	-	✓	-	✓	-	-	-
Kimera-VIO	✓	✓	-	✓	-	-	-
ORB-SLAM3	✓	✓	-	✓	-	-	-

- Wide range of systems available for visual-inertial research
- Our group's: OpenVINS which is a ***feature complete filter system*** for use on resource constrained platforms and as an odometry frontend

OpenVINS

- An open platform for **VINS research** (OpenVINS) which achieves state-of-the-art performance
- On manifold sliding window Kalman filter with modular ***type system*** for state management (gtsam inspired)
- Detailed **documentation** and derivations to support researchers using the codebase: https://docs.openvins.com/



https://github.com/rpng/open_vins

[1] Geneva, P., Eckenhoff, K., Lee, W., Yang, Y. and Huang, G., 2020, November. OpenVINS: A research platform for visual-inertial estimation. In Proc. of the IEEE International Conference on Robotics and Automation, Paris, France.

OpenVINS – Key Features

- Sliding window visual-inertial MSCKF
- Modular covariance type system
- Comprehensive documentation and derivations
- Extendable visual-inertial simulator
  - On manifold SE(3) b-spline
  - Arbitrary number of cameras
  - Arbitrary sensor rate
  - Automatic feature generation
- Five different feature representations
- Environmental SLAM feature
  - OpenCV ARUCO tag SLAM features
  - Sparse feature SLAM features
- Calibration of sensor intrinsics and extrinsics
  - Camera to IMU transform
  - Camera to IMU time offset
  - Camera intrinsics
- Visual tracking support
  - Monocular / Stereo / Binocular cameras
  - KLT or descriptor based
- First-Estimate Jacobians for consistent estimation
- Static IMU initialization
- Zero velocity detection and updates
- Out of the box dataset evaluation on:
  - EurocMav
  - TUM-VI
  - UZH-FPV Drone Racing
  - KAIST Urban Driving
- Extensive evaluation suite:
  - ATE, RPE, NEES, RMSE
  - Timing evaluation and plotting
- Codebase extensions:
  - ov_secondary – Secondary pose graph with loop-closure
  - ov_maplab – Multi-session mapping and offline optimization
  - vicon2gt – Groundtruth gen. for VIO dataset evaluation

OpenVINS – Key Features

- Sliding window visual-inertial MSCKF
- Modular covariance type system
- Comprehensive documentation and derivations
- **Extendable visual-inertial simulator**
  - **On manifold SE(3) b-spline**
  - **Arbitrary number of cameras**
  - **Arbitrary sensor rate**
  - **Automatic feature generation**
- Five different feature representations
- Environmental SLAM feature
  - OpenCV ARUCO tag SLAM features
  - Sparse feature SLAM features
- Calibration of sensor intrinsics and extrinsics
  - Camera to IMU transform
  - Camera to IMU time offset
  - Camera intrinsics
- Visual tracking support
  - Monocular / Stereo / Binocular cameras
  - KLT or descriptor based
- First-Estimate Jacobians for consistent estimation
- Static IMU initialization
- Zero velocity detection and updates
- Out of the box dataset evaluation on:
  - EurocMav
  - TUM-VI
  - UZH-FPV Drone Racing
  - KAIST Urban Driving
- Extensive evaluation suite:
  - ATE, RPE, NEES, RMSE
  - Timing evaluation and plotting
- Codebase extensions:
  - ov_secondary – Secondary pose graph with loop-closure
  - ov_maplab – Multi-session mapping and offline optimization
  - vicon2gt – Groundtruth gen. for VIO dataset evaluation

OpenVINS – Key Features

- Sliding window visual-inertial MSCKF
- Modular covariance type system
- Comprehensive documentation and derivations
- **Extendable visual-inertial simulator**
  - **On manifold SE(3) b-spline**
  - **Arbitrary number of cameras**
  - **Arbitrary sensor rate**
  - **Automatic feature generation**
- Five different feature representations
- Environmental SLAM feature
  - OpenCV ARUCO tag SLAM features
  - Sparse feature SLAM features
- **Calibration of sensor intrinsics and extrinsics**
  - **Camera to IMU transform**
  - **Camera to IMU time offset**
  - **Camera intrinsics**
- Visual tracking support
  - Monocular / Stereo / Binocular cameras
  - KLT or descriptor based
- First-Estimate Jacobians for consistent estimation
- Static IMU initialization
- Zero velocity detection and updates
- Out of the box dataset evaluation on:
  - EurocMav
  - TUM-VI
  - UZH-FPV Drone Racing
  - KAIST Urban Driving
- Extensive evaluation suite:
  - ATE, RPE, NEES, RMSE
  - Timing evaluation and plotting
- Codebase extensions:
  - ov_secondary – Secondary pose graph with loop-closure
  - ov_maplab – Multi-session mapping and offline optimization
  - vicon2gt – Groundtruth gen. for VIO dataset evaluation

OpenVINS – Key Features

- Sliding window visual-inertial MSCKF
- Modular covariance type system
- Comprehensive documentation and derivations
- **Extendable visual-inertial simulator**
  - **On manifold SE(3) b-spline**
  - **Arbitrary number of cameras**
  - **Arbitrary sensor rate**
  - **Automatic feature generation**
- Five different feature representations
- Environmental SLAM feature
  - OpenCV ARUCO tag SLAM features
  - Sparse feature SLAM features
- **Calibration of sensor intrinsics and extrinsics**
  - **Camera to IMU transform**
  - **Camera to IMU time offset**
  - **Camera intrinsics**
- Visual tracking support
  - Monocular / Stereo / Binocular cameras
  - KLT or descriptor based
- First-Estimate Jacobians for consistent estimation
- Static IMU initialization
- Zero velocity detection and updates
- **Out of the box dataset evaluation on:**
  - **EurocMav**
  - **TUM-VI**
  - **UZH-FPV Drone Racing**
  - **KAIST Urban Driving**
- **Extensive evaluation suite:**
  - **ATE, RPE, NEES, RMSE**
  - **Timing evaluation and plotting**
- **Codebase extensions:**
  - **ov_secondary – Secondary pose graph with loop-closure**
  - **ov_maplab – Multi-session mapping and offline optimization**
  - **vicon2gt – Groundtruth gen. for VIO dataset evaluation**

Lots of great features directly "out-of-the-box" to enable research and practical deployment!

Wide Range of VINS Datasets

Dataset	Year	Enviroment	Groundtruth	Sync?	Other Sensors	Platform
EuRoC MAV	2016	Indoor Structured	6D Vicon 3D Laser Tracker	Yes	-	UAV
NCLT	2016	Large-scale Outdoor Structured	RTK+SLAM	No	LiDAR Ladybug Wheel Encoders	Rover
TUM VI	2018	Indoor / Outdoor Structured	Partial 6D Vicon	Yes	-	Handheld
TUM VI RS	2019	Indoor Structured	6D Vicon	Yes	Rolling and Global Shutter	Handheld
UZH-FPV Drone	2019	Indoor / Outdoor	3D Laser Tracker	No	Event Camera	UAV
KAIST Urban	2019	Large-scale Outdoor Structured	RTK+SLAM	Yes	LiDAR Wheel Encoders FOG	Car
Blackbird	2020	Simulated Indoor Structured	6D Vicon	Yes	Motor Tachometers Depth Segmentation	UAV
Newer College Datset	2020	Outdoor Structured	ICP to Prior Map	No	LiDAR Depth	Handheld
MADMAX	2021	Outdoor Mars Enviroment	5D RTK	Yes	LiDAR Depth	Rover
NTU VIRAL	2021	Outdoor Structured	3D Laser Tracker	No	LiDAR UWB	UAV

Wide Range of VINS Datasets

Dataset	Year	Enviroment	Groundtruth	Sync?	Other Sensors	Platform
EuRoC MAV	2016	Indoor Structured	6D Vicon 3D Laser Tracker	Yes	-	UAV
NCLT	2016	Large-scale Outdoor Structured	RTK+SLAM	No	LiDAR Ladybug Wheel Encoders	Rover
TUM VI	2018	Indoor / Outdoor Structured	Partial 6D Vicon	Yes	-	Handheld
TUM VI RS	2019	Indoor Structured	6D Vicon	Yes	Rolling and Global Shutter	Handheld
UZH-FPV Drone	2019	Indoor / Outdoor	3D Laser Tracker	No	Event Camera	UAV
KAIST Urban	2019	Large-scale Outdoor Structured	RTK+SLAM	Yes	LiDAR Wheel Encoders FOG	Car
Blackbird	2020	Simulated Indoor Structured	6D Vicon	Yes	Motor Tachometers Depth Segmentation	UAV
Newer College Datset	2020	Outdoor Structured	ICP to Prior Map	No	LiDAR Depth	Handheld
MADMAX	2021	Outdoor Mars Enviroment	5D RTK	Yes	LiDAR Depth	Rover
NTU VIRAL	2021	Outdoor Structured	3D Laser Tracker	No	LiDAR UWB	UAV

!

Metrics for Evaluation

- Quality of groundtruth remains challenging in realworld datasets
- High quality metrics allow for **fair comparison** of different algorithms
- Open Source Toolboxes:
  - evo - https://github.com/MichaelGrupp/evo
  - rpg_trajectory_evaluation - https://github.com/uzh-rpg/rpg_trajectory_evaluation
  - ov_eval - https://github.com/rpng/open_vins

- Absolute Trajectory Error (ATE)

$$e_{ATE} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{K}\sum_{k=1}^{K}||\mathbf{x}_{k,i} - \hat{\mathbf{x}}_{k,i}^{+}||_2^2}$$

- Relative Pose Error (RPE) **[recommended]**

$$\tilde{\mathbf{x}}_r = \mathbf{x}_k - \mathbf{x}_{k+d_i}$$ ← *Relative pose for trajectory segments*

$$e_{rpe,d_i} = \frac{1}{D_i}\sum_{k=1}^{D_i}||\tilde{\mathbf{x}}_r - \hat{\tilde{\mathbf{x}}}_r||_2^2$$

- Normalized Estimation Error Squared (NEES)

$$e_{nees,k} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_{k,i} - \hat{\mathbf{x}}_{k,i})^\top \mathbf{P}_{k,i}^{-1}(\mathbf{x}_{k,i} - \hat{\mathbf{x}}_{k,i})$$

[1] Zhang and Scaramuzza. "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry." International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.

Metrics for Evaluation

- Quality of groundtruth remains challenging in realworld datasets
- High quality metrics allow for **fair comparison** of different algorithms
- Open Source Toolboxes:
  - evo - https://github.com/MichaelGrupp/evo
  - rpg_trajectory_evaluation - https://github.com/uzh-rpg/rpg_trajectory_evaluation
  - ov_eval - https://github.com/rpng/open_vins

Fast evaluation tool with additional recording and timing utilities!

- Absolute Trajectory Error (ATE)

$$e_{ATE} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{K}\sum_{k=1}^{K}||\mathbf{x}_{k,i} - \hat{\mathbf{x}}_{k,i}^{+}||_2^2}$$

- Relative Pose Error (RPE) **[recommended]**

$$\tilde{\mathbf{x}}_r = \mathbf{x}_k - \mathbf{x}_{k+d_i}$$

Relative pose for trajectory segments

$$e_{rpe,d_i} = \frac{1}{D_i}\sum_{k=1}^{D_i}||\tilde{\mathbf{x}}_r - \hat{\tilde{\mathbf{x}}}_r||_2^2$$

- Normalized Estimation Error Squared (NEES)

$$e_{nees,k} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_{k,i} - \hat{\mathbf{x}}_{k,i})^\top \mathbf{P}_{k,i}^{-1}(\mathbf{x}_{k,i} - \hat{\mathbf{x}}_{k,i})$$

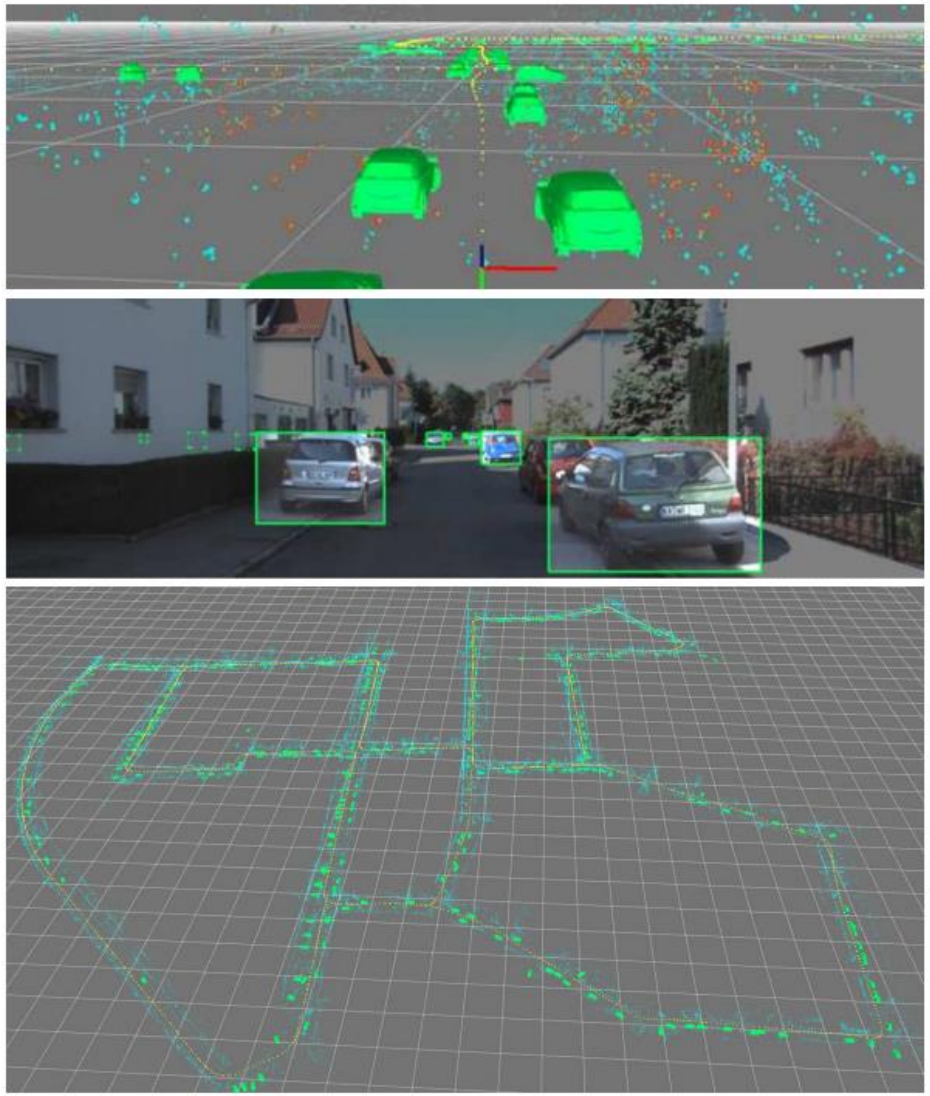
[1] Zhang and Scaramuzza. "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry." International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.

Outline – Conclusion

- Future Directions
- Conclusion

Where Next?

- **Practicality:** Many challenges to widely deploying VINS (e.g. known calib etc.)
- **Robustness:** Moving environment, long-term seasons, sensor variances
- **Semantic Understanding:** Object-wise and uncertain network classifications
- **Computational:** Real-time robotic systems (low-cost IoT devices, latency, etc.)
- **Aided-INS:** Incorporating additional sensors (e.g. event, thermal, etc.)
- **Cooperative:** Multi-robot systems (measurement selection, distributed, scalability, etc.)
- **Dynamics:** Integrate robot dynamics

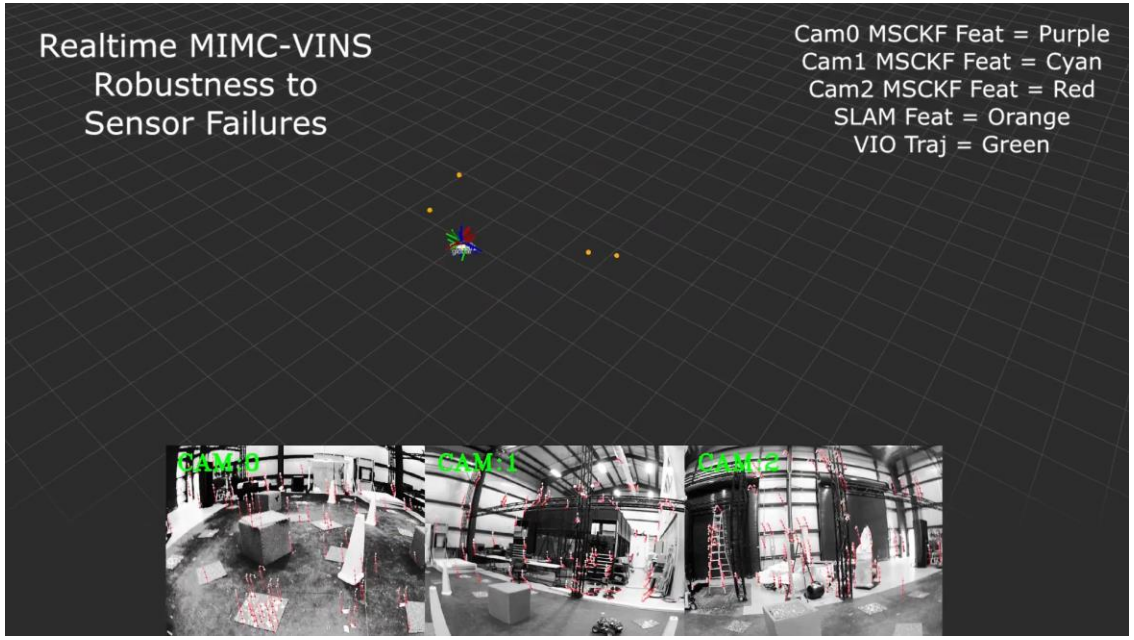


[1] Huang, Guoquan. "Visual-inertial navigation: A concise review." 2019 international conference on robotics and automation (ICRA). IEEE, 2019.
[photo] Dong, Jingming, Xiaohan Fei, and Stefano Soatto. "Visual-inertial-semantic scene representation for 3D object detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

Summary & Thanks!

- This presentation:
  - Introduced background on traditional VINS estimators
  - Discussed challenges in designing VINS algorithms
  - Presented summary of current open-sourced codebases, datasets, and evaluation tools

- Contact information:
  - Patrick Geneva (@goldbattle github)
  - pgeneva@udel.edu
  - https://pgeneva.com/



Please checkout OpenVINS!

https://github.com/rpng/open_vins

https://docs.openvins.com/