UNMANNED SYSTEMS LAB

TEXAS A&M UNIVERSITY
J. Mike Walker '66 Department of Mechanical Engineering

Online Multi-IMU Calibration Using Visual-Inertial Odometry

Jacob Hartzer

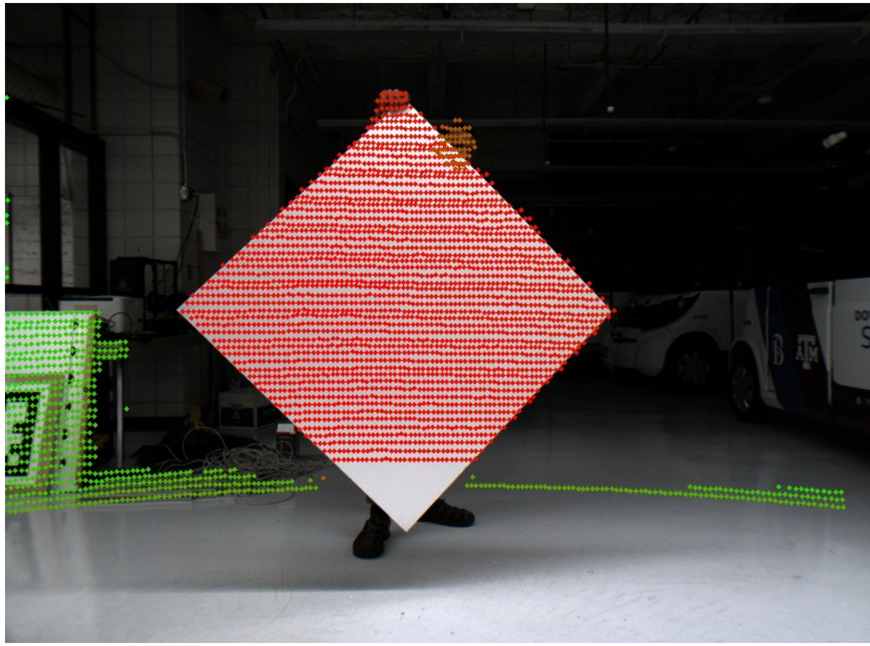
SDF and MFI 2023

mfi23 Bonn

SDF

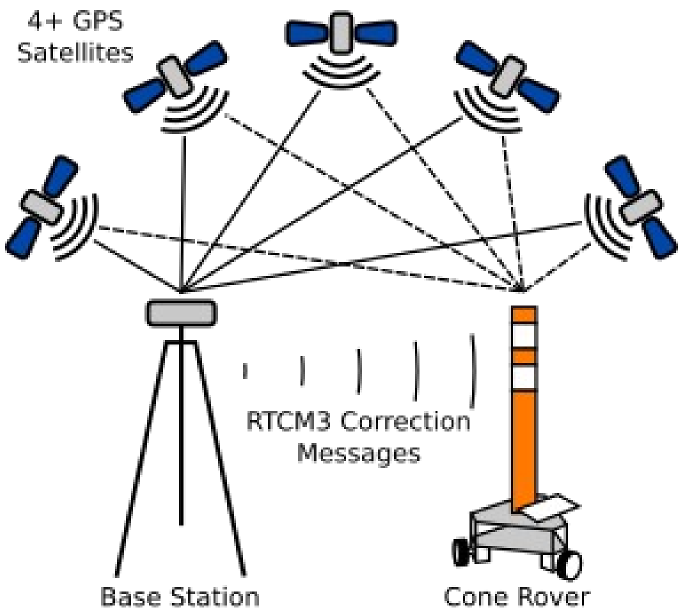
Unmanned Systems Lab

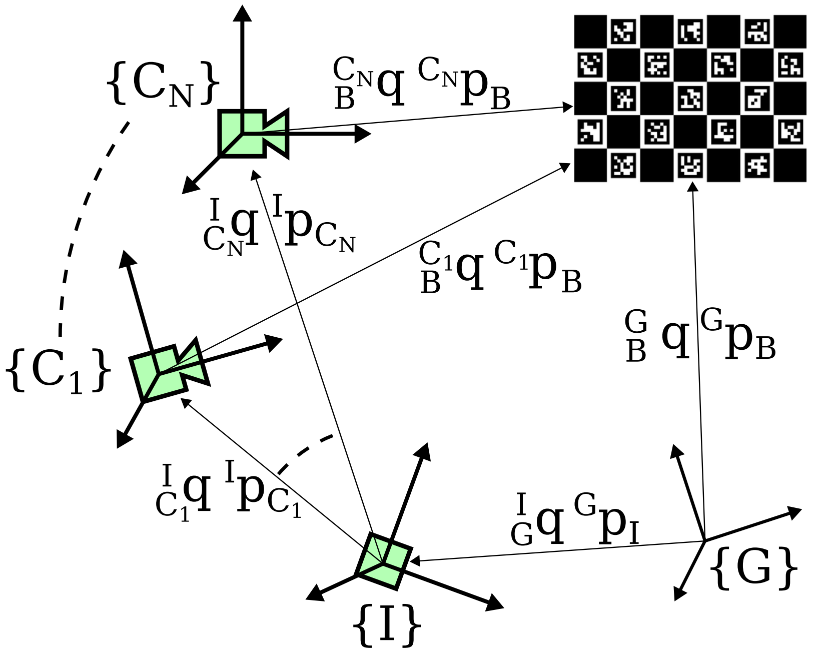
- On/Off Road Autonomy
- Semantic Segmentation
- Multi-Sensor Fusion Calibration

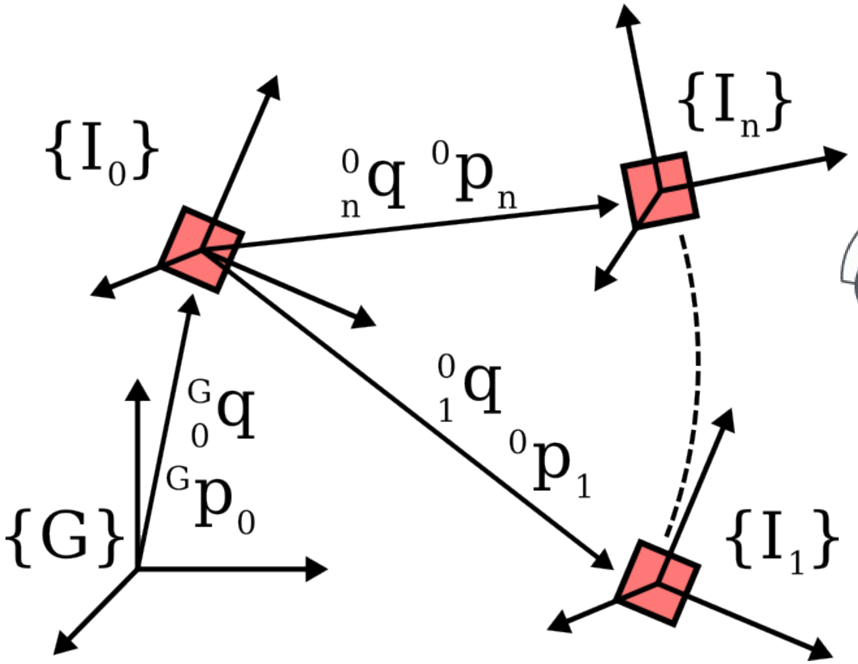


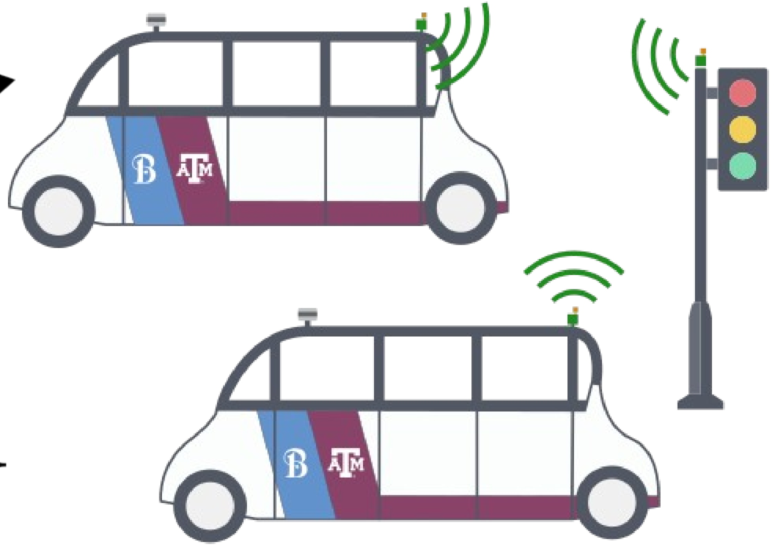
Our Work - Calibration

- Utilize a wide variety of sensors and environments
- RTK GPS in highway applications and truth data
- Ultra-Wideband for cooperative localization
- Multi-IMU, Multi-Camera fusion



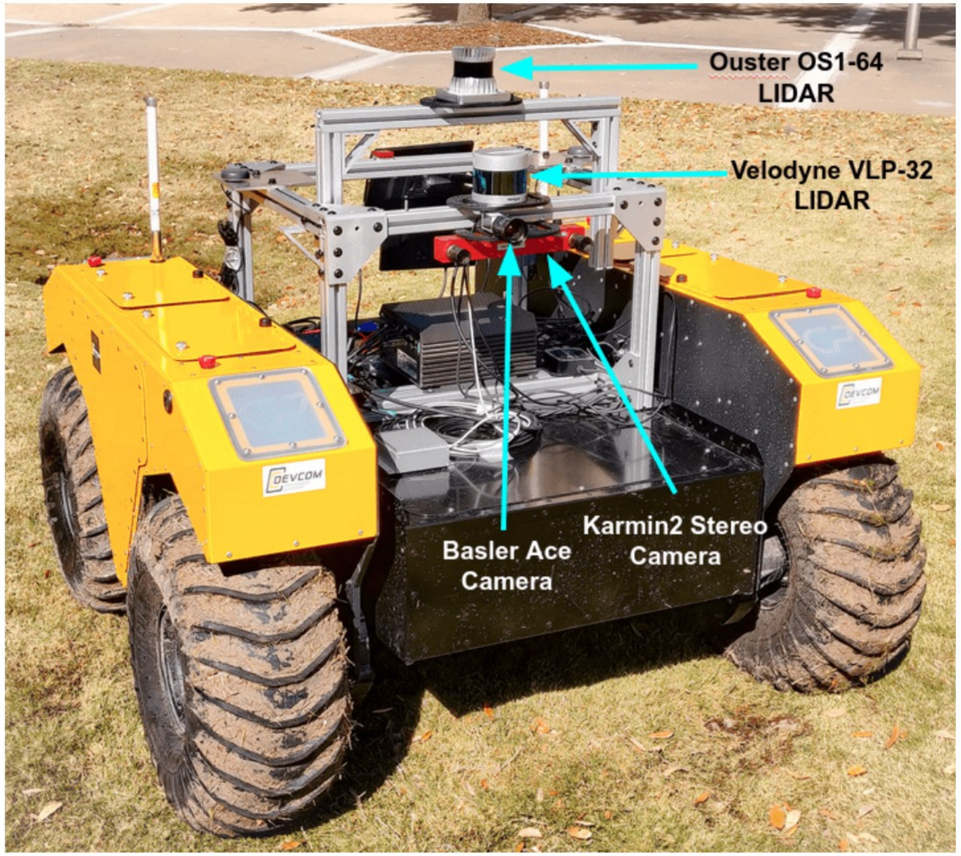






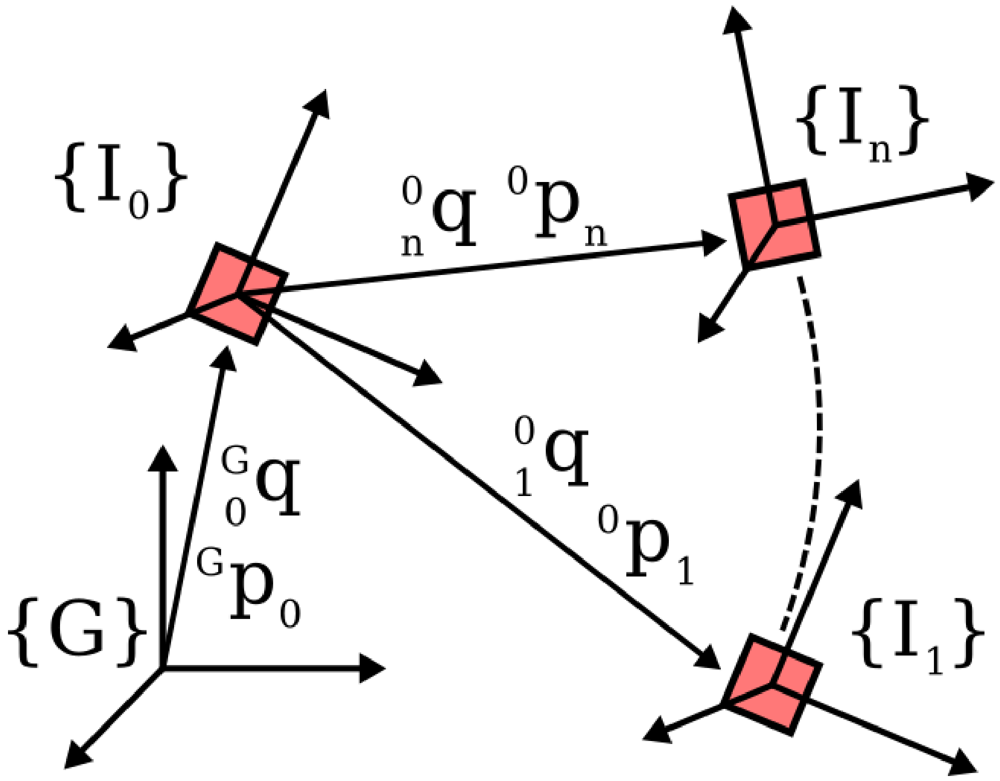
The Challenge of Calibration

- Sensors do not work well when uncalibrated
- Ideally would like to calibrate sensors *online*
- Re-calibrating on the fly is more robust
- Proving stability and consistency is <u>hard</u>
- Sensor flexibility makes it even harder



Problem Statement

- Develop online calibration estimates for multiple IMU
- Retain flexibility in the number of IMUs
- Can handle large baseline distances
- Does not require synchronization
- Can utilize differing quality of sensors
- Robust against dropouts



Typical EKF Prediction

- EKF Prediction step typically utilizes specific forces and angular rates
  - Single IMU measurements can be used directly
  - Multiple IMU measurements are fused into virtual measurement
  - Federated filters architectures run multiple IMU filters simultaneously

IMU Error Models

- Start with and simulate typical gyroscope and accelerometer errors
- For this work, we assume scaled and orthogonal measurements

$$\begin{bmatrix} \tilde{a}_x \\ \tilde{a}_y \\ \tilde{a}_z \end{bmatrix} = \begin{bmatrix} S_{ax} & 0 & 0 \\ 0 & S_{ay} & 0 \\ 0 & 0 & S_{az} \end{bmatrix} \begin{bmatrix} 1 & \alpha_{a1} & \alpha_{a2} \\ \alpha_{a3} & 1 & \alpha_{a4} \\ \alpha_{a5} & \alpha_{a6} & 1 \end{bmatrix} \left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} + \begin{bmatrix} b_{ax} \\ b_{ay} \\ b_{az} \end{bmatrix} \right) + \begin{bmatrix} n_{ax} \\ n_{ay} \\ n_{az} \end{bmatrix}$$

$$\begin{bmatrix} \tilde{\omega}_x \\ \tilde{\omega}_y \\ \tilde{\omega}_z \end{bmatrix} = \begin{bmatrix} S_{\omega x} & 0 & 0 \\ 0 & S_{\omega y} & 0 \\ 0 & 0 & S_{\omega z} \end{bmatrix} \begin{bmatrix} 1 & \alpha_{\omega 1} & \alpha_{\omega 2} \\ \alpha_{\omega 3} & 1 & \alpha_{\omega 4} \\ \alpha_{\omega 5} & \alpha_{\omega 6} & 1 \end{bmatrix} \left(\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + \begin{bmatrix} b_{\omega x} \\ b_{\omega y} \\ b_{\omega z} \end{bmatrix} \right) + \begin{bmatrix} n_{\omega x} \\ n_{\omega y} \\ n_{\omega z} \end{bmatrix}$$

Typical EKF Prediction

- Typical EKF prediction uses an estimate of angular rate and acceleration
- Measurements must be synchronized or fused
- Only requires retention of 9 states (position, velocity, orientation)

State:

$$\boldsymbol{x}_b = \begin{bmatrix} \boldsymbol{p} & \boldsymbol{v} & {}^G_B q \end{bmatrix}$$

Propagation:

$$ {}^G\dot{\boldsymbol{p}}_I = {}^G\boldsymbol{v}_I $$

$$ {}^G\dot{\boldsymbol{v}}_I = {}^G\boldsymbol{a} $$

$$ {}^I_G\dot{q} = \frac{1}{2}\Omega(\omega)\, {}^I_G q $$

Where:

$$\Omega(\boldsymbol{\omega}) = \begin{bmatrix} -[\boldsymbol{\omega}]_\times & \boldsymbol{\omega} \\ \boldsymbol{\omega}^T & 0 \end{bmatrix}$$

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

IMU Measurement Model

- A set of desynchronized, separated IMU cannot be easily fused
- For generalized IMU measurements, consider the following model

$$\underbrace{\boldsymbol{a}_m}_{\text{Measured Acceleration}} = \overbrace{\boldsymbol{a}}^{\text{True Acceleration}} + \underbrace{\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{r})}_{\text{Centrifugal Acceleration}} + \overbrace{\boldsymbol{\alpha} \times \boldsymbol{r}}^{\text{Euler Acceleration}}$$

"Full State" EKF Prediction

- To accommodate measurement functions, we must retain 18 states
- Predictions are now separate from IMU measurements
- Measurements can now provide **calibration** or **non-calibration** updates

State:

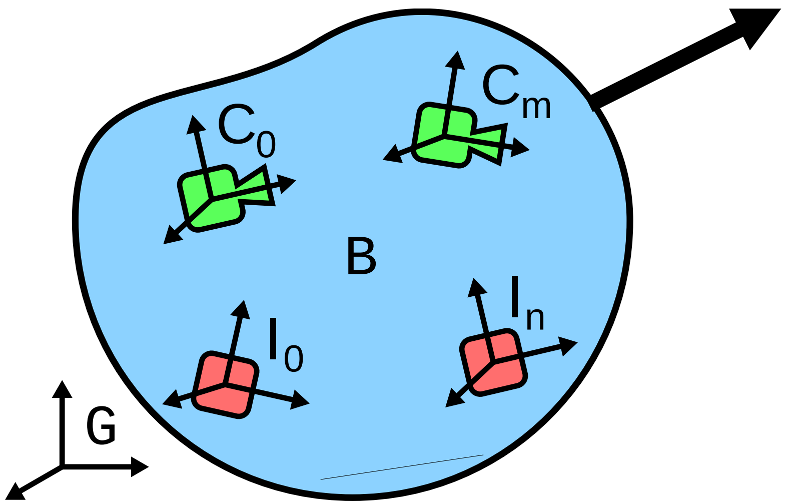
$$\boldsymbol{x}_b = \begin{bmatrix} \boldsymbol{p} & \boldsymbol{v} & \boldsymbol{a} & {}^G_B q & \boldsymbol{\omega} & \boldsymbol{\alpha} \end{bmatrix}$$

Propagation:

$$\boldsymbol{f}\left(\hat{\boldsymbol{x}}^-_{k-1}\right) = \begin{bmatrix} \hat{\boldsymbol{p}}^-_{k-1} & + & \hat{\boldsymbol{v}}^-_{k-1}\Delta t \\ \hat{\boldsymbol{v}}^-_{k-1} & + & (\hat{\boldsymbol{a}}^-_{k-1} - \mathcal{C}({}^G_B q^-_{k-1})\boldsymbol{g})\Delta t \\ \hat{\boldsymbol{a}}^-_{k-1} & & \\ {}^G_B q^-_{k-1} & \otimes & q(\hat{\boldsymbol{\omega}}_{k-1}\Delta t) \\ \hat{\boldsymbol{\omega}}^-_{k-1} & + & \hat{\boldsymbol{\alpha}}^-_{k-1}\Delta t \\ \hat{\boldsymbol{\alpha}}^-_{k-1} & & \end{bmatrix}$$

IMU Non-Calibration Update

- IMUs generate measurements
- Observation matrix is generated from state information
- Only provides update to body state



$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{a}_m \\ \boldsymbol{\omega}_m \end{bmatrix}$$

$$\boldsymbol{h}(\hat{\boldsymbol{x}}_b) = \begin{bmatrix} \mathcal{C}({}^B_{I_i}q)^T \left(\mathcal{C}({}^G_B q)^T \hat{\boldsymbol{a}} + \hat{\boldsymbol{\alpha}} \times {}^B\hat{\boldsymbol{p}}_{I_i} + \hat{\boldsymbol{\omega}} \times \hat{\boldsymbol{\omega}} \times {}^B\hat{\boldsymbol{p}}_{I_i} \right) \\ \mathcal{C}({}^B_{I_i}q)^T \mathcal{C}({}^G_B q)^T \hat{\boldsymbol{\omega}} \end{bmatrix}$$

$$\boldsymbol{H}_b = \begin{bmatrix} \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \frac{\partial \boldsymbol{a}_m}{\partial \hat{\boldsymbol{a}}} & \boldsymbol{0}_{3\times3} & \frac{\partial \boldsymbol{a}_m}{\partial \hat{\boldsymbol{\omega}}} & \frac{\partial \boldsymbol{a}_m}{\partial \hat{\boldsymbol{\alpha}}} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \frac{\partial \boldsymbol{\omega}_m}{\partial \hat{\boldsymbol{\omega}}} & \boldsymbol{0}_{3\times3} \end{bmatrix}$$

$$\frac{\partial \boldsymbol{a}_m}{\partial \hat{\boldsymbol{a}}} = \mathcal{C}({}^B_{I_i}q)^T \mathcal{C}({}^G_B q)^T$$

$$\frac{\partial \boldsymbol{a}_m}{\partial \hat{\boldsymbol{\omega}}} = \mathcal{C}({}^B_{I_i}q)^T \left([\hat{\boldsymbol{\omega}}]_\times \left[{}^B\hat{\boldsymbol{p}}_{I_i}\right]_\times^T + \left[\hat{\boldsymbol{\omega}} \times {}^B\hat{\boldsymbol{p}}_{I_i}\right]_\times^T \right)$$

$$\frac{\partial \boldsymbol{a}_m}{\partial \hat{\boldsymbol{\alpha}}} = \mathcal{C}({}^B_{I_i}q)^T \left[{}^B\hat{\boldsymbol{p}}_{I_i}\right]_\times^T$$

$$\frac{\partial \boldsymbol{\omega}_m}{\partial \hat{\boldsymbol{\omega}}} = \left[\mathcal{C}({}^B_{I_i}q)^T \mathcal{C}({}^G_B q)^T \hat{\boldsymbol{\omega}}\right]_\times$$

IMU Calibration Update

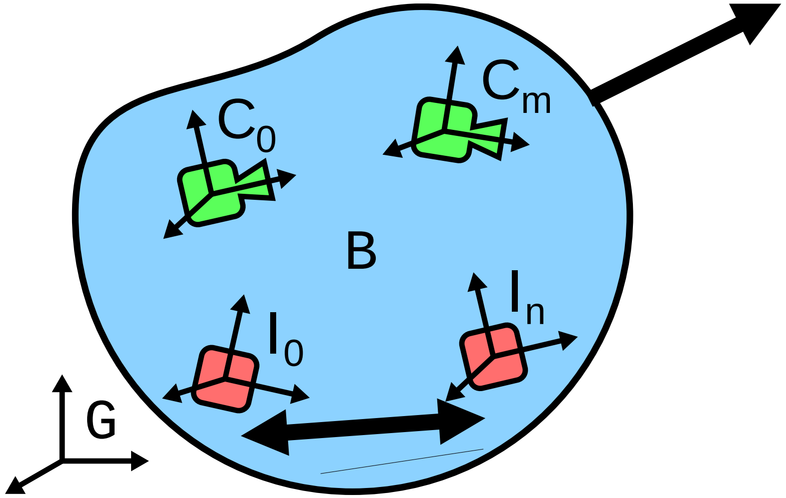
$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{a}_m \\ \boldsymbol{\omega}_m \end{bmatrix}$$

- IMUs generate measurements
- Observation matrix is generated from state information
- Provides update to:
  - Body State
  - IMU Calibration States

$$\boldsymbol{x}_{I_N} = \begin{bmatrix} {}^{B}\boldsymbol{p}_{I_i} & {}^{B}_{I_i}q & \boldsymbol{b}_a & \boldsymbol{b}_\omega \end{bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_b & \boldsymbol{H}_{i_0} & \cdots & \boldsymbol{H}_{i_N} \end{bmatrix}$$

$$\boldsymbol{H}_i = \begin{bmatrix} \frac{\partial \boldsymbol{a}_m}{\partial\, {}^{B}\hat{\boldsymbol{p}}_{I_i}} & \frac{\partial \boldsymbol{a}_m}{\partial\, {}^{B}_{I_i}\hat{q}} & \mathbf{I}_3 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \frac{\partial \boldsymbol{\omega}_m}{\partial\, {}^{B}_{I_i}\hat{q}} & \mathbf{0}_{3\times3} & \mathbf{I}_3 \end{bmatrix}$$



$$\frac{\partial \boldsymbol{a}_m}{\partial\, {}^{B}\hat{\boldsymbol{p}}_{I_i}} = \mathcal{C}({}^{B}_{I_i}\hat{q})^T \left([\boldsymbol{\alpha}]_\times + [\hat{\omega}]_\times [\hat{\omega}]_\times \right)$$

$$\frac{\partial \boldsymbol{a}_m}{\partial\, {}^{B}_{I_i}\hat{q}} = \left[\mathcal{C}({}^{B}_{I_i}\hat{q})^T \left((\hat{\boldsymbol{\alpha}} + \hat{\boldsymbol{\omega}} \times \hat{\boldsymbol{\omega}}) \times {}^{B}p_{I_i} + \mathcal{C}({}^{G}_{B}\hat{q})^T \hat{\boldsymbol{a}} \right) \right]_\times$$

$$\frac{\partial \boldsymbol{\omega}_m}{\partial\, {}^{B}_{I_i}\hat{q}} = \left[\mathcal{C}({}^{B}_{I_i}\hat{q})^T \mathcal{C}({}^{G}_{B}\hat{q})^T \hat{\boldsymbol{\omega}} \right]_\times$$

Kalman Update

- Typical extended Kalman update procedure is utilized for both update types

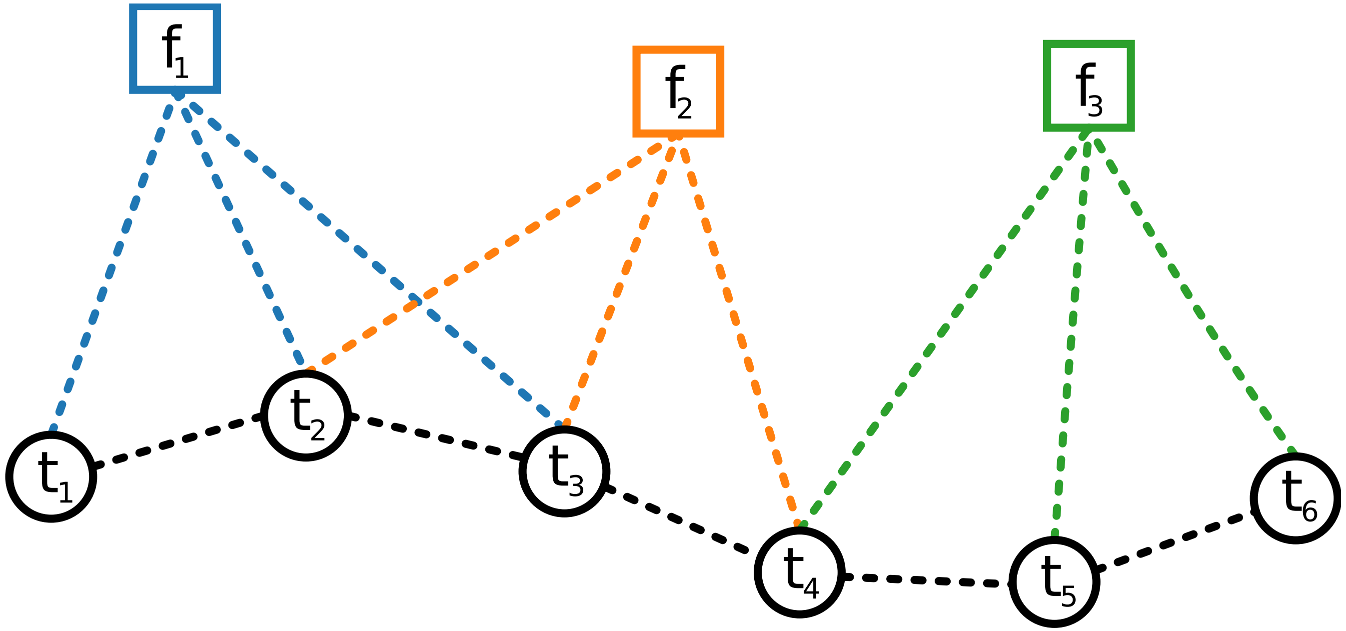
$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T(\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}))$$

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_{k|k-1}$$

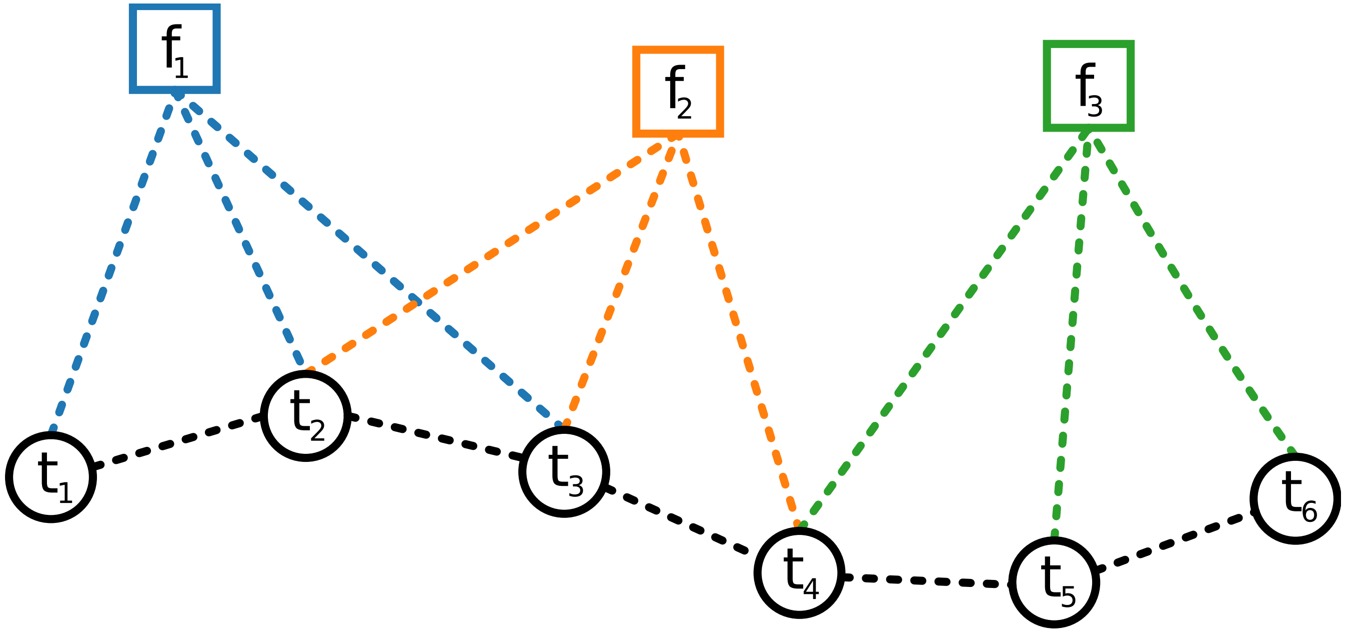
Observability Limitations

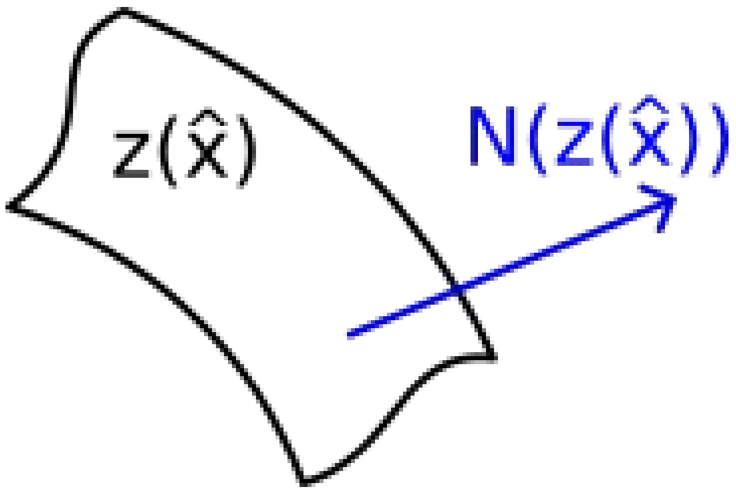
- Even with idealized trajectories, not all IMU states are observable
  - E.G. the filter cannot determine all IMU biases from motion alone
- Therefore, this sub-filter is paired with Visual-Odometry
- A MSCKF was selected for its flexibility



MSCKF Implementation

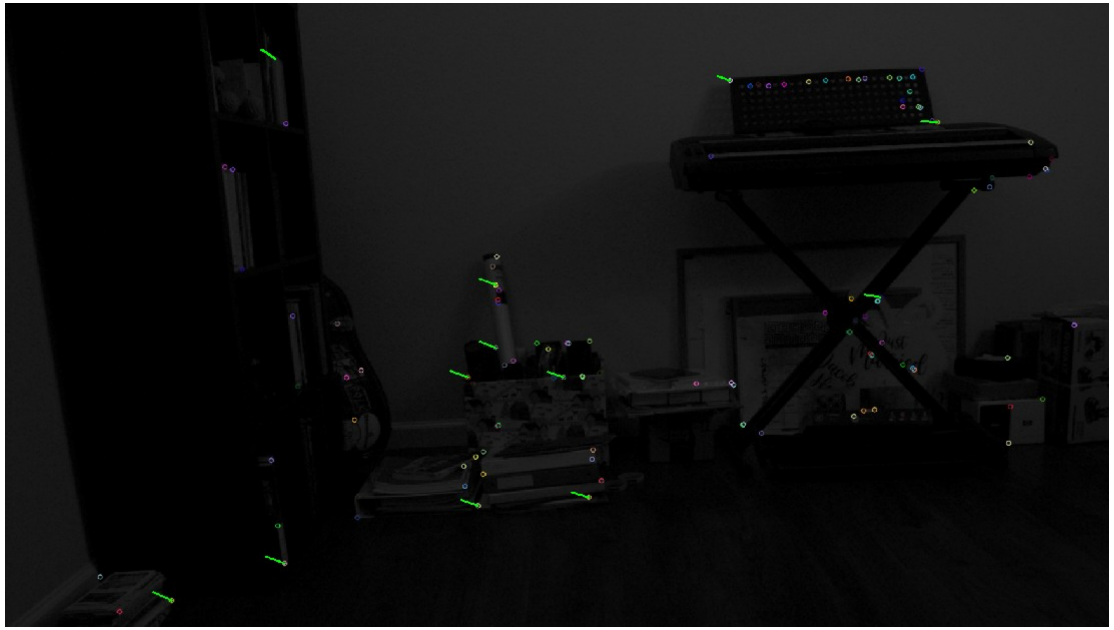
- The MSCKF is utilizes augmented states to retain state history
  - The body frame is used for state augmentation instead of IMU frames
  - The body frame can be fixed to an IMU frame, for simplicity
- Features are tracked and utilized in a state update
  - Nullspace projection is still utilized





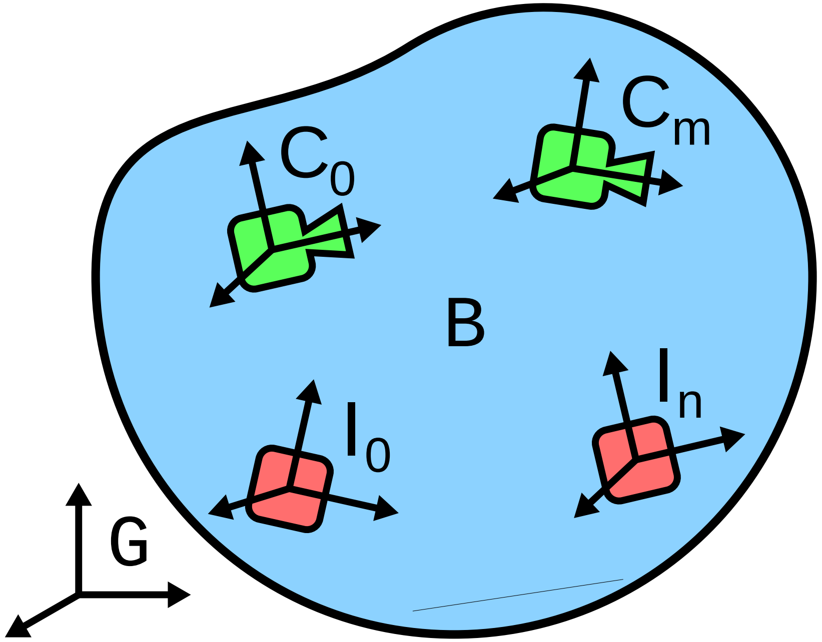
MSCKF Detectors & Matchers

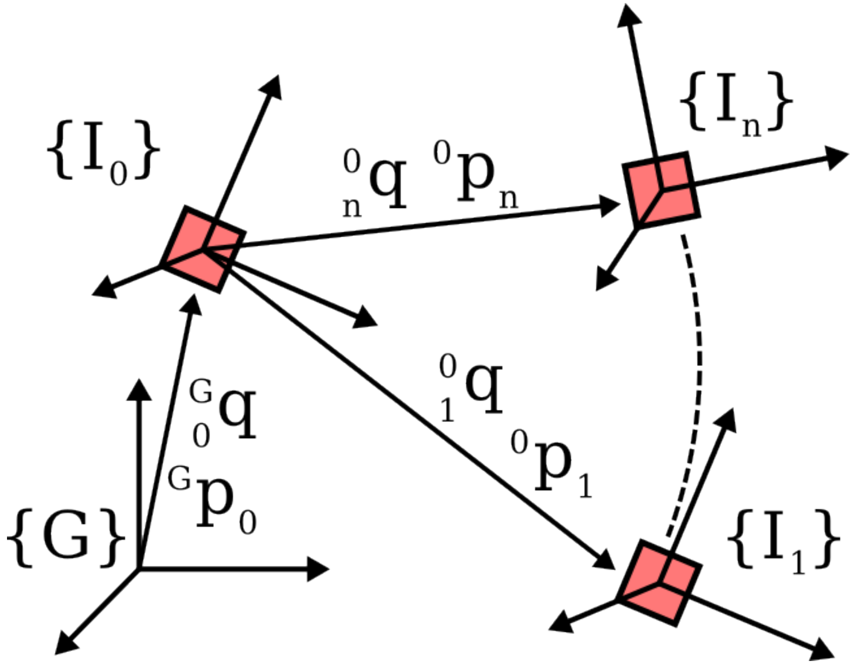
- For a MSCKF, the visual front-end is flexible
- In this work, OpenCV is used for the front end (FAST, ORB, FLANN)
- We are considering testing deep-learned features as well



EKF-CAL Package

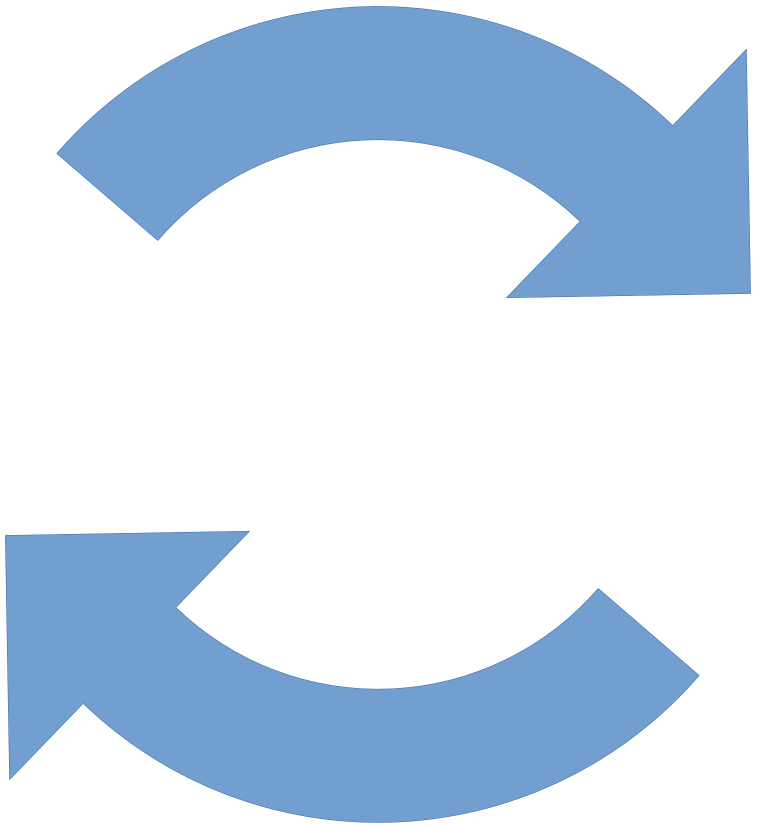
- EKF-CAL is a flexible MSCKF-based sensor calibration package
- Inputs are YAML based and compatible with ROS parameter declarations
- Inherently multi-sensor (IMU, Camera, and GPS soon)
- Developed with integrated testing and Monte Carlo simulation in mind





Typical Development

Need performance / Real-Time

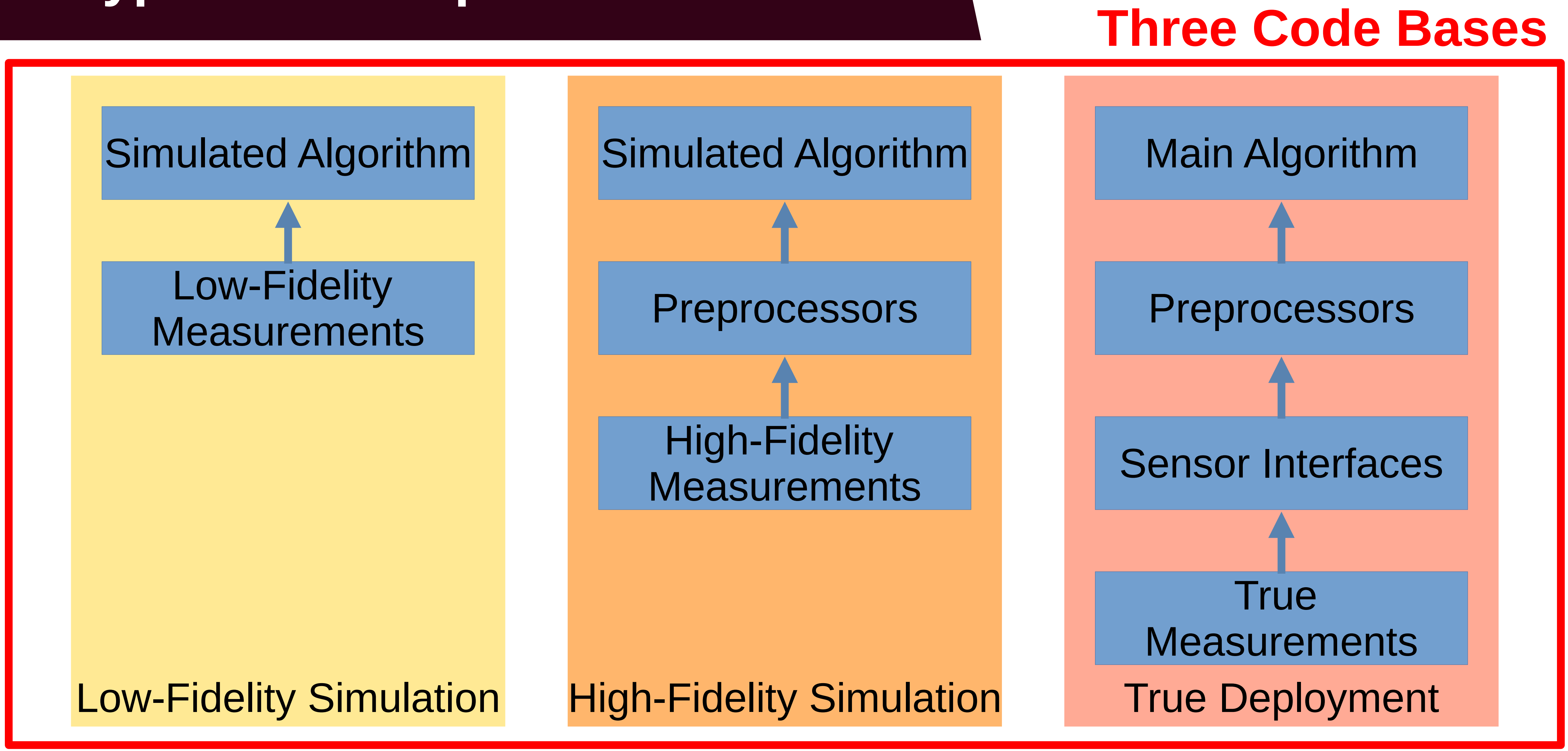


Develop Algorithm in scripted language (MatLab, Julia, etc.)

Deploy code in compiled language (C++, Rust, etc.)

Find a bug / invalid assumptions

Typical Development

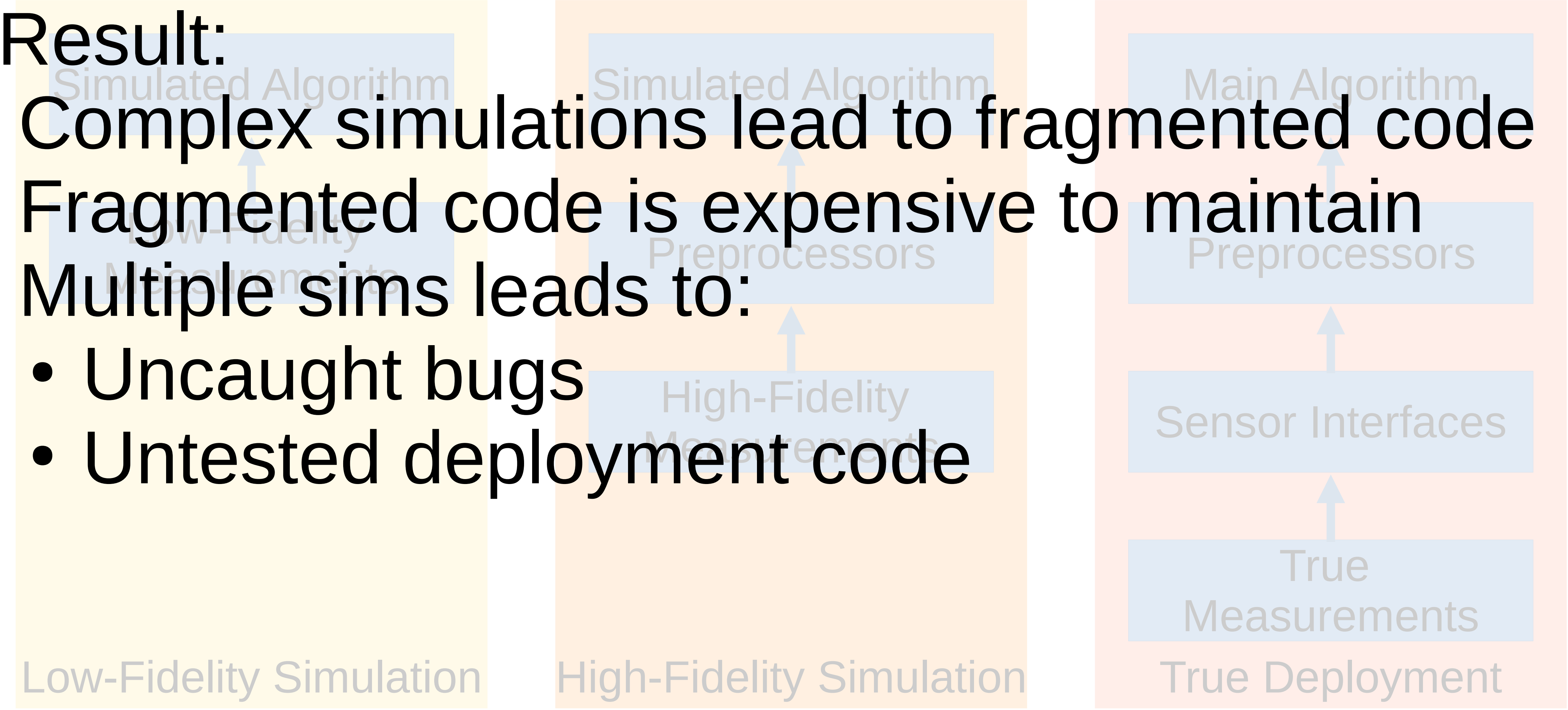


Typical Development

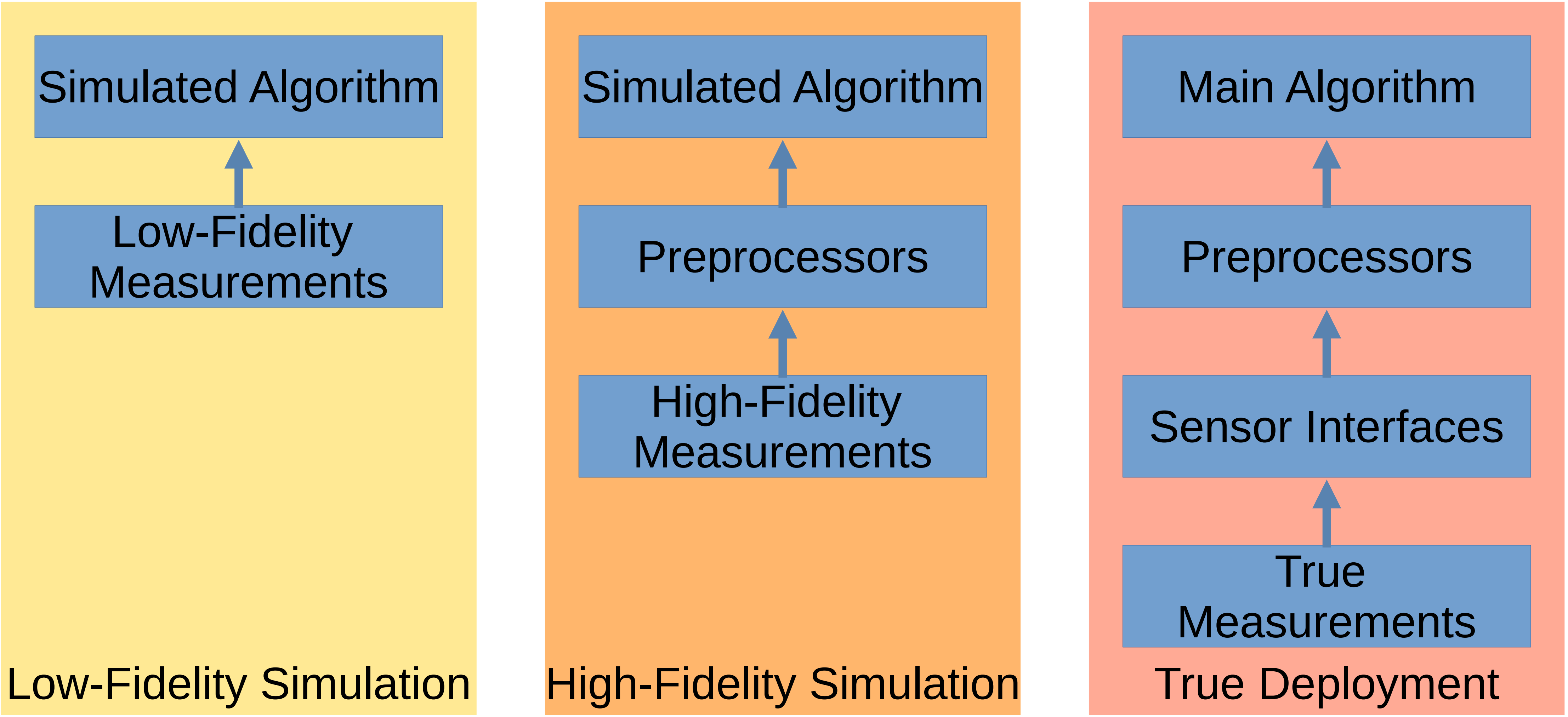
Three Code Bases

Result:

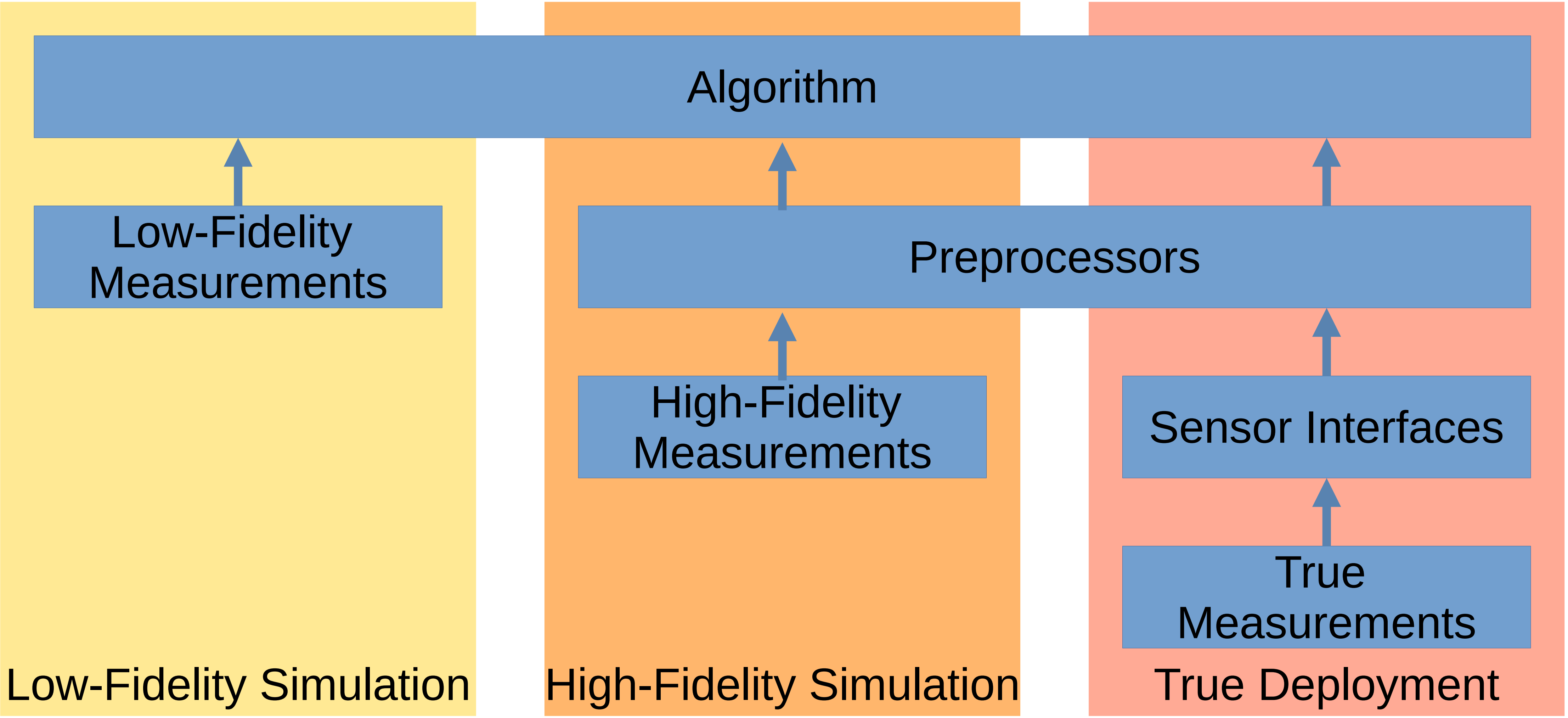
- Complex simulations lead to fragmented code
- Fragmented code is expensive to maintain
- Multiple sims leads to:
  - Uncaught bugs
  - Untested deployment code



Typical Development

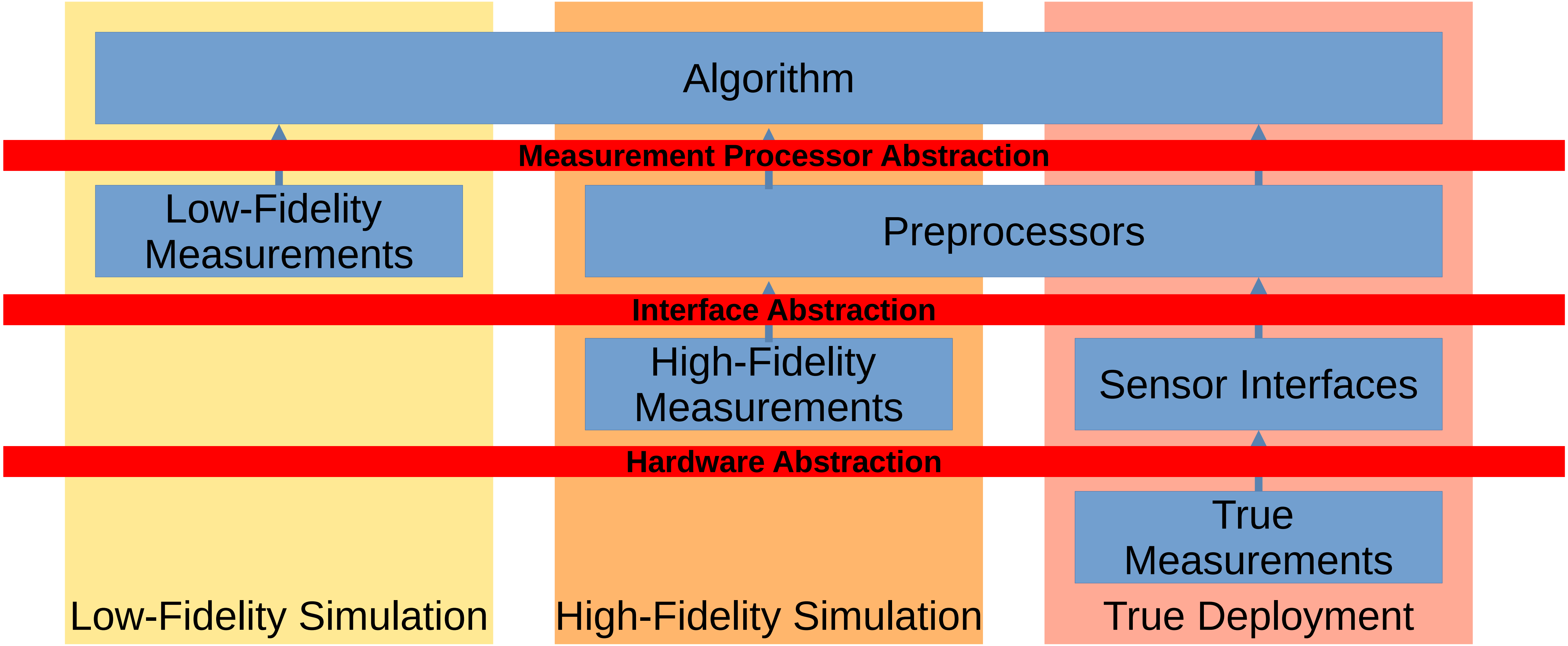


Utilized Solution



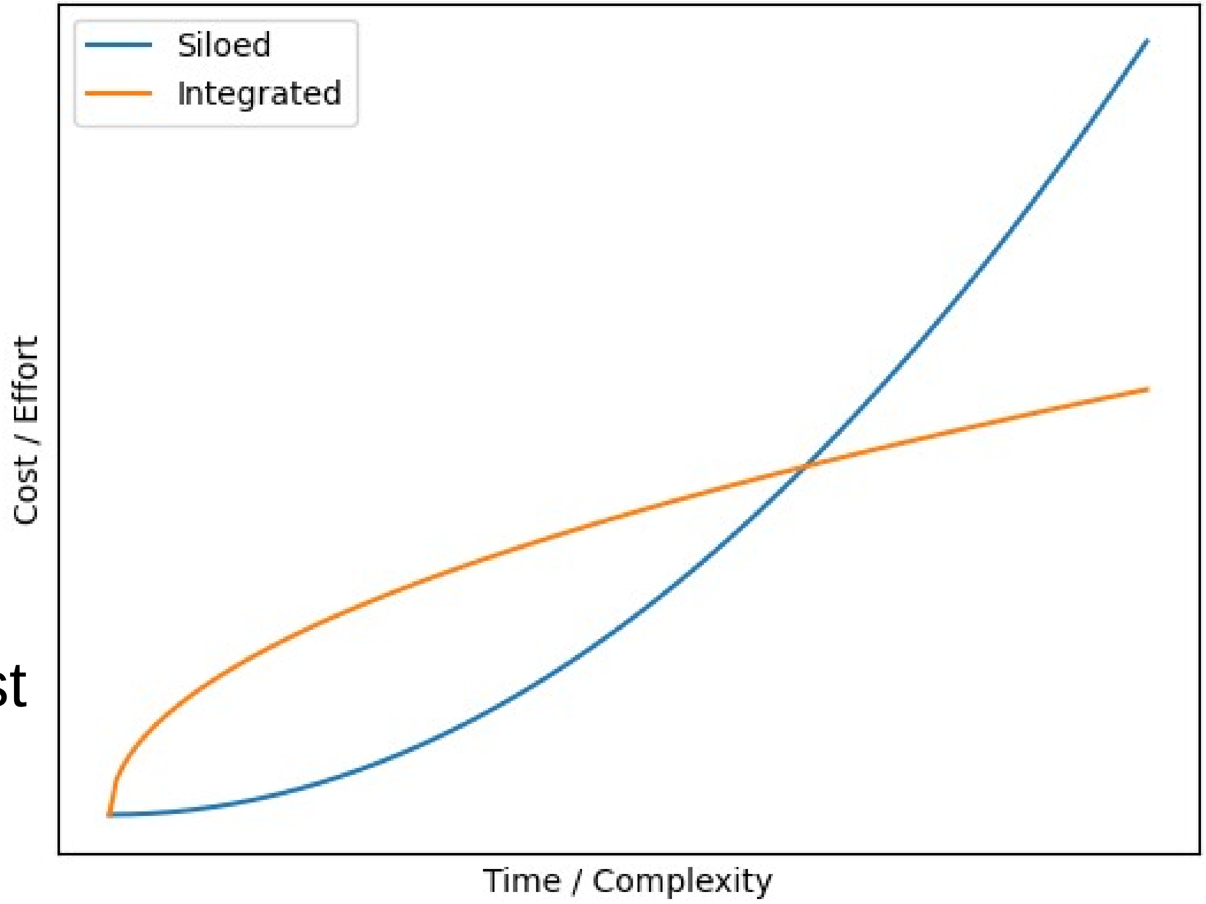
Utilized Solution

Abstraction Layers!



Integrated

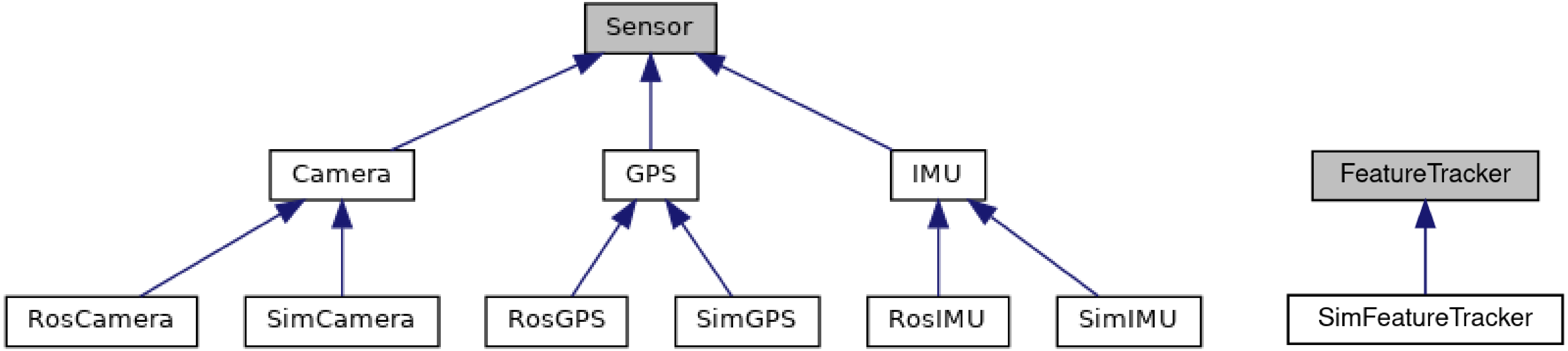
Higher initial cost



Pays off in long-run

Sensor Abstraction

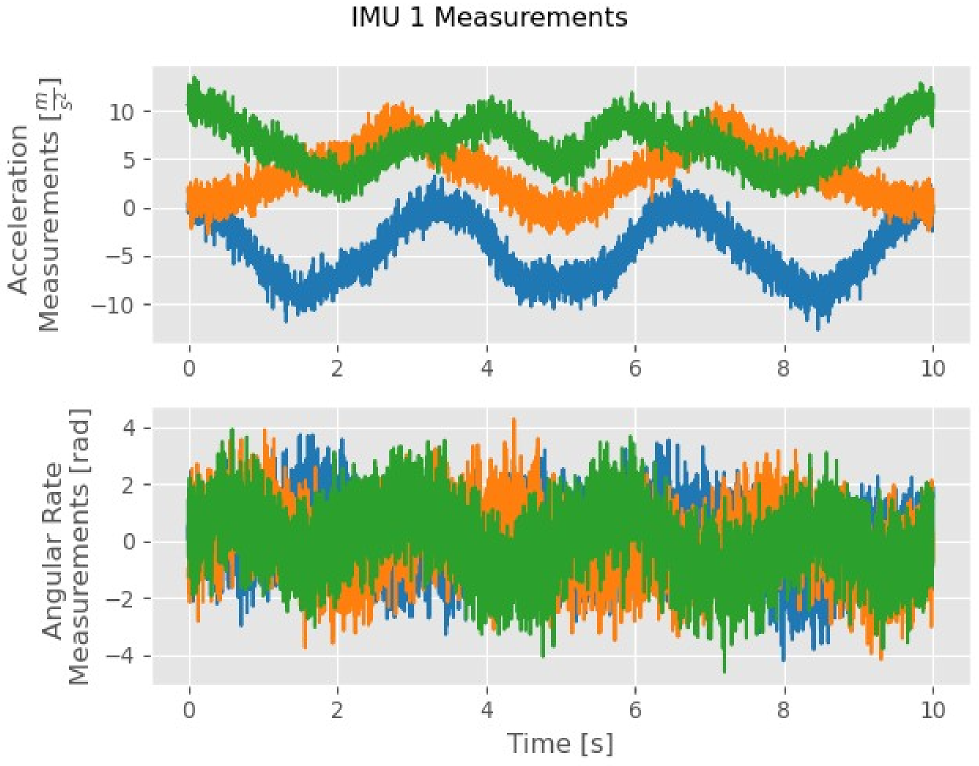
- Sensors call updates to filter / algorithm
  - Utilize real or simulated sensor messages
- Feature tracker utilizes camera measurements
  - High-Fidelity Simulation provides ray-traced images
  - Low-Fidelity Simulation provides “pre-tracked features”



Abstraction Models: IMU

$$\begin{bmatrix} \tilde{a}_x \\ \tilde{a}_y \\ \tilde{a}_z \end{bmatrix} = \begin{bmatrix} S_{ax} & 0 & 0 \\ 0 & S_{ay} & 0 \\ 0 & 0 & S_{az} \end{bmatrix} \begin{bmatrix} 1 & \alpha_{a1} & \alpha_{a2} \\ \alpha_{a3} & 1 & \alpha_{a4} \\ \alpha_{a5} & \alpha_{a6} & 1 \end{bmatrix} \left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} + \begin{bmatrix} b_{ax} \\ b_{ay} \\ b_{az} \end{bmatrix} \right) + \begin{bmatrix} n_{ax} \\ n_{ay} \\ n_{az} \end{bmatrix}$$

$$\begin{bmatrix} \tilde{\omega}_x \\ \tilde{\omega}_y \\ \tilde{\omega}_z \end{bmatrix} = \begin{bmatrix} S_{\omega x} & 0 & 0 \\ 0 & S_{\omega y} & 0 \\ 0 & 0 & S_{\omega z} \end{bmatrix} \begin{bmatrix} 1 & \alpha_{\omega 1} & \alpha_{\omega 2} \\ \alpha_{\omega 3} & 1 & \alpha_{\omega 4} \\ \alpha_{\omega 5} & \alpha_{\omega 6} & 1 \end{bmatrix} \left(\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + \begin{bmatrix} b_{\omega x} \\ b_{\omega y} \\ b_{\omega z} \end{bmatrix} \right) + \begin{bmatrix} n_{\omega x} \\ n_{\omega y} \\ n_{\omega z} \end{bmatrix}$$



Abstraction Models: Camera

$$x_n = \frac{{}^C\boldsymbol{p}_x f_x + c_x}{w\,{}^C\boldsymbol{p}_z} - 1$$

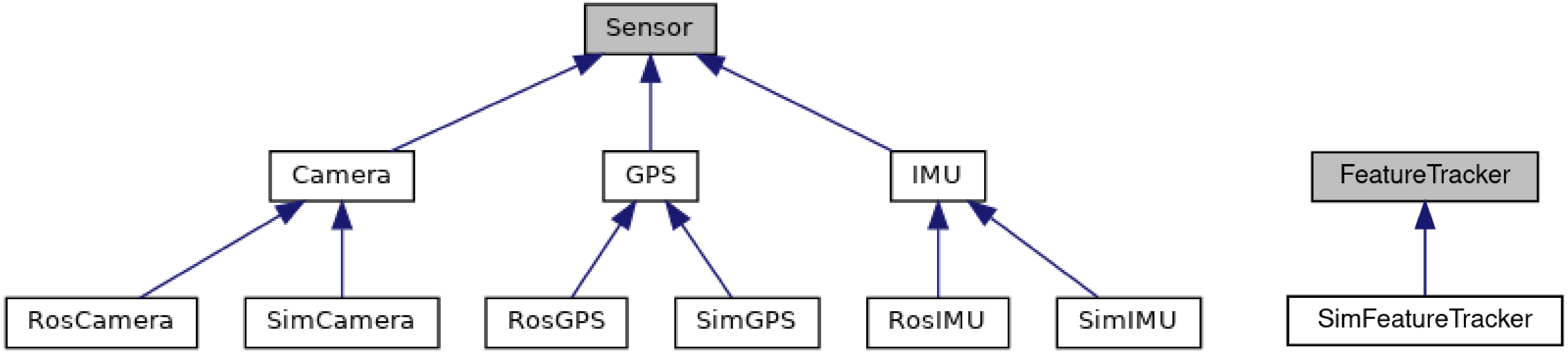
$$y_n = \frac{{}^C\boldsymbol{p}_y f_y + c_y}{h\,{}^C\boldsymbol{p}_z} - 1$$

$$r^2 = x_n^2 + y_n^2$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} \cdot \underbrace{\begin{bmatrix} 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \\ 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \end{bmatrix}}_{\text{Radial}} + \underbrace{\begin{bmatrix} 2p_1 x_n y_n + p_2 \cdot (r^2 + 2x_n^2) \\ p_1 \cdot (r^2 + 2y_n^2) + 2p_2 x_n y_n \end{bmatrix}}_{\text{Tangential}}$$

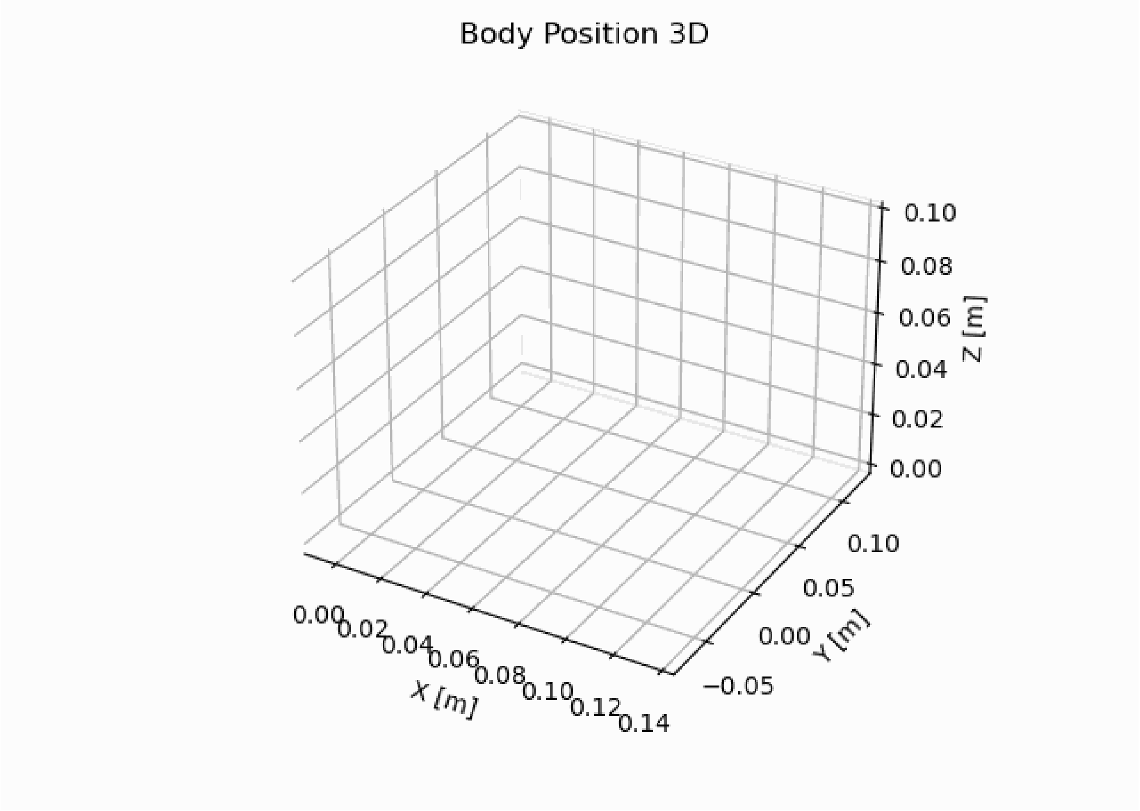
Abstraction - Benefits

- Improves accuracy of simulation
- Catches more bugs earlier
- Reduces rework (no code divergence)
- More beneficial unit testing
- Robust Monte Carlo testing



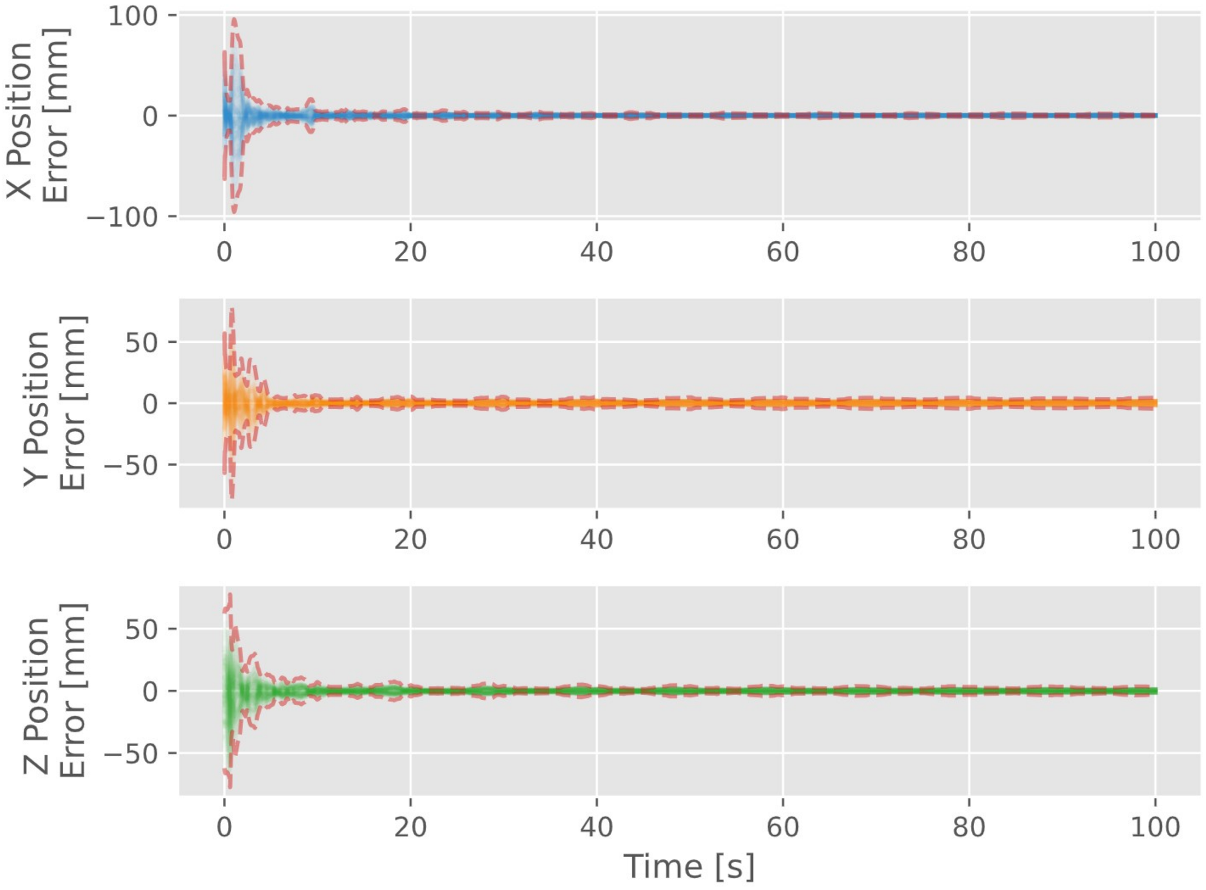
Monte Carlo Testing

- With fast enough simulations, we can run thousands of example datasets
- Random initialization and measurement errors are inserted
- Utilizing abstractions increases confidence of filter stability

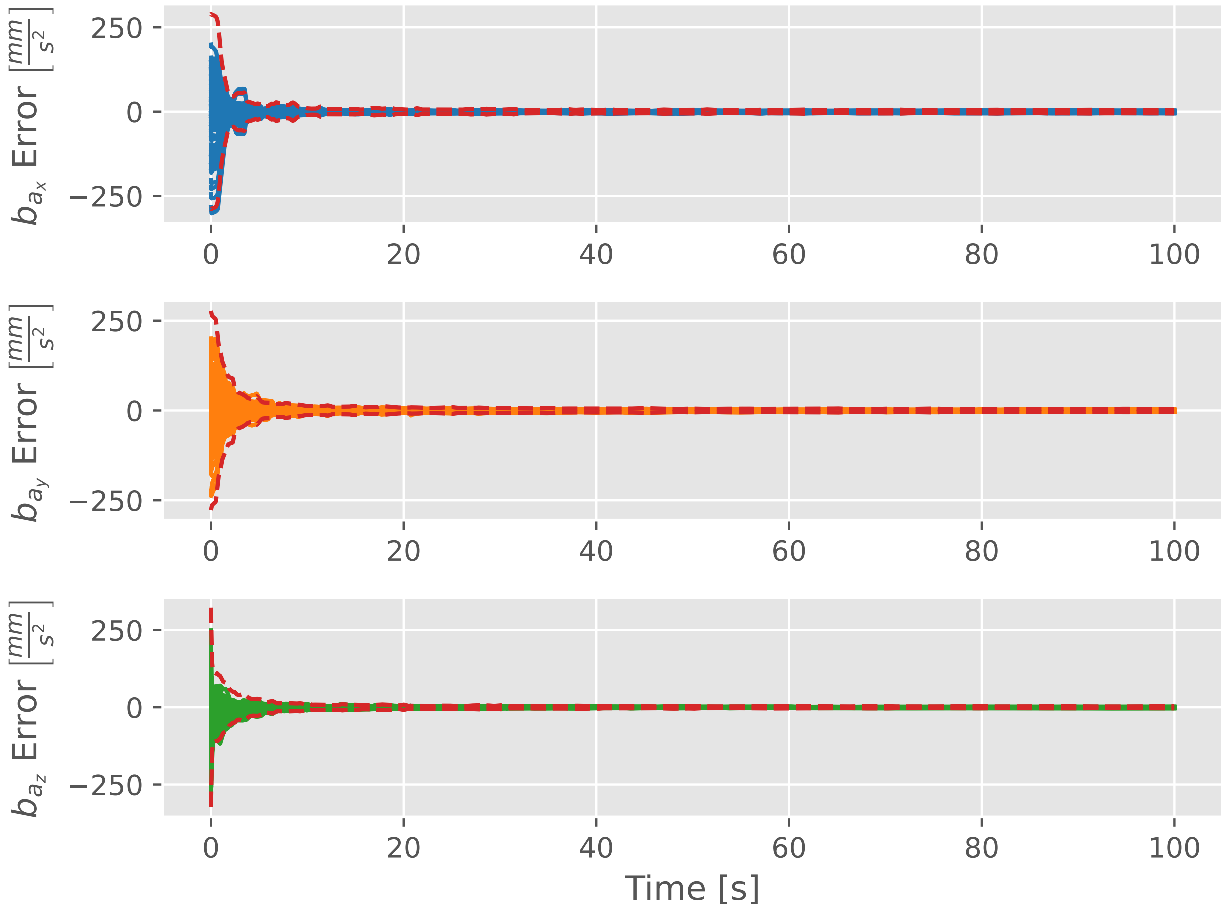


Monte Carlo Testing

Secondary IMU Position Convergence

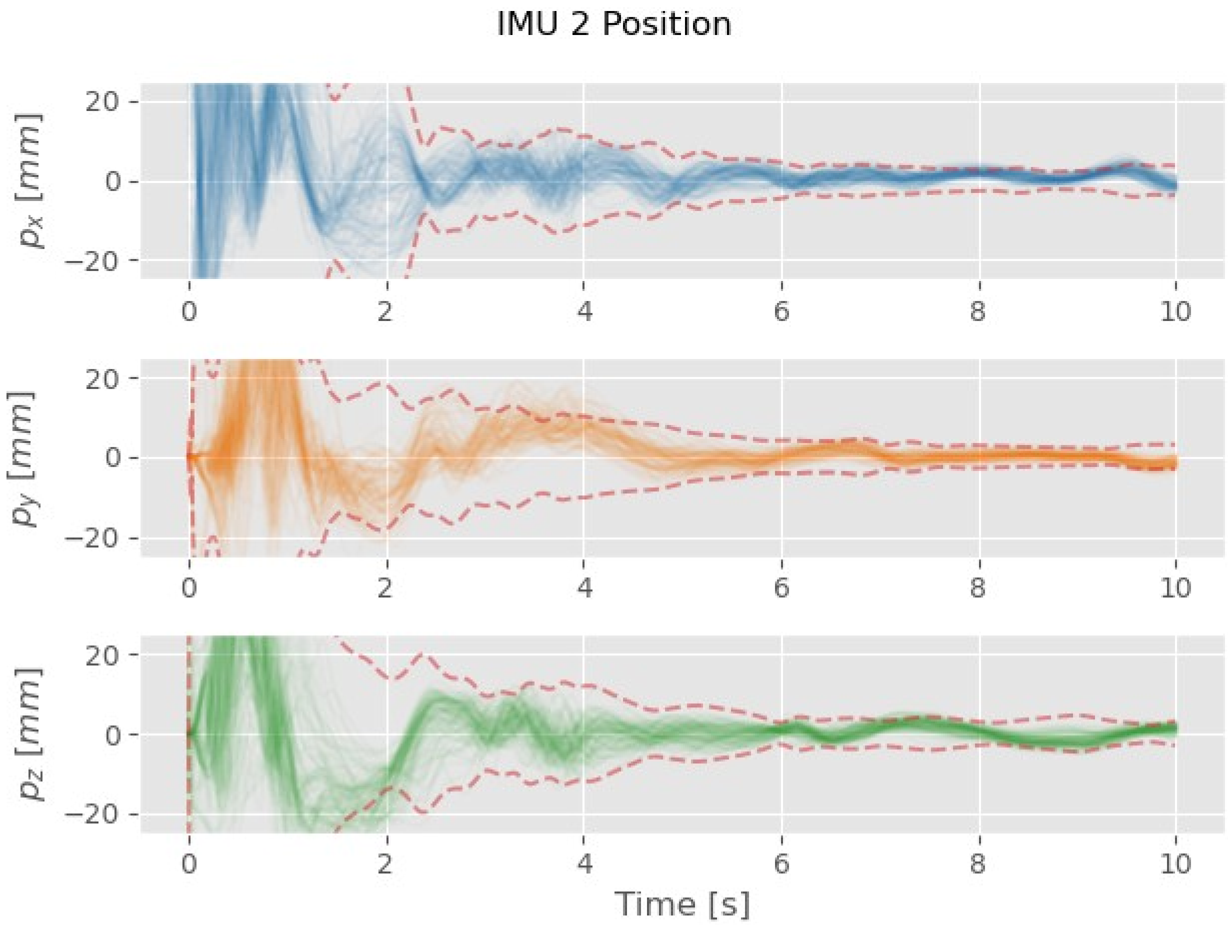


Secondary IMU Bias Convergence



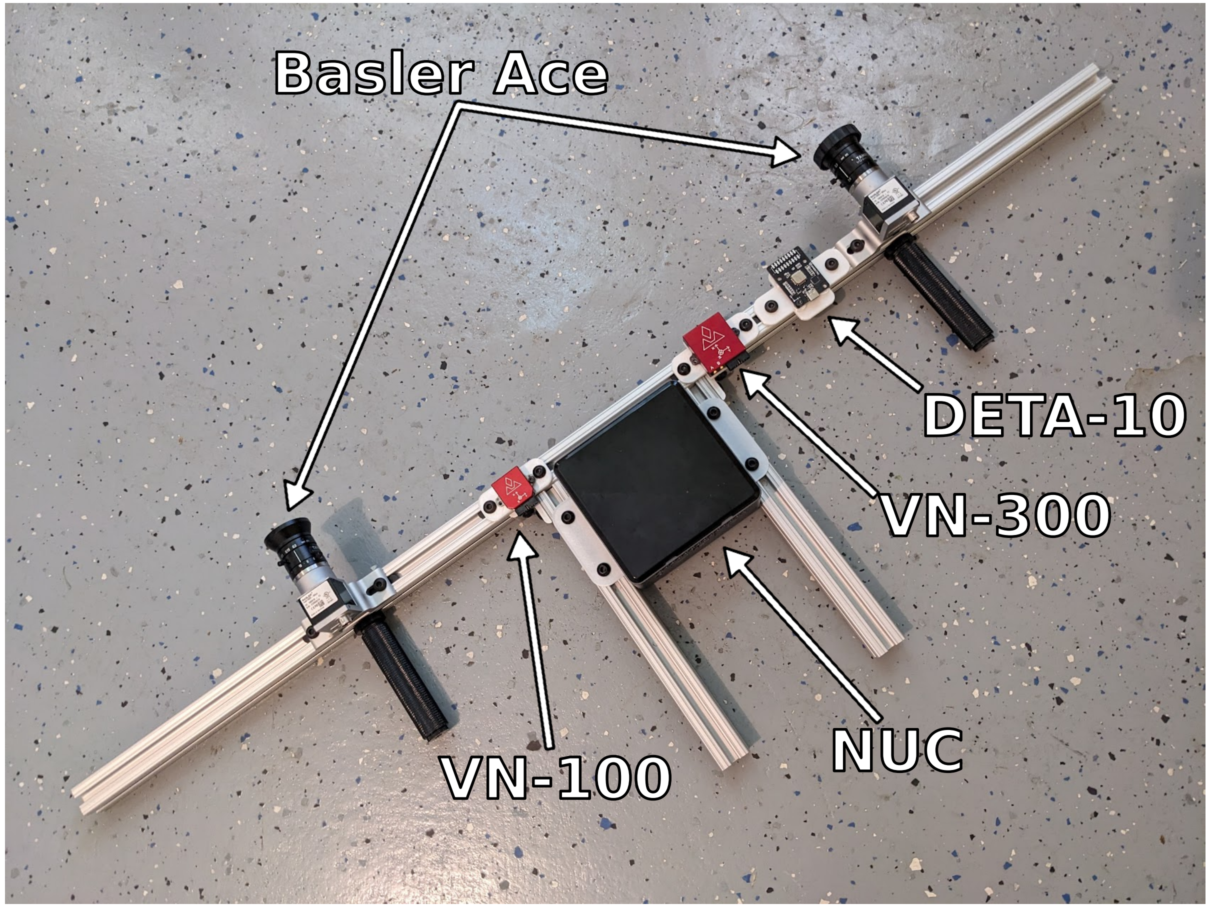
Monte Carlo Testing

Secondary IMU Position Convergence



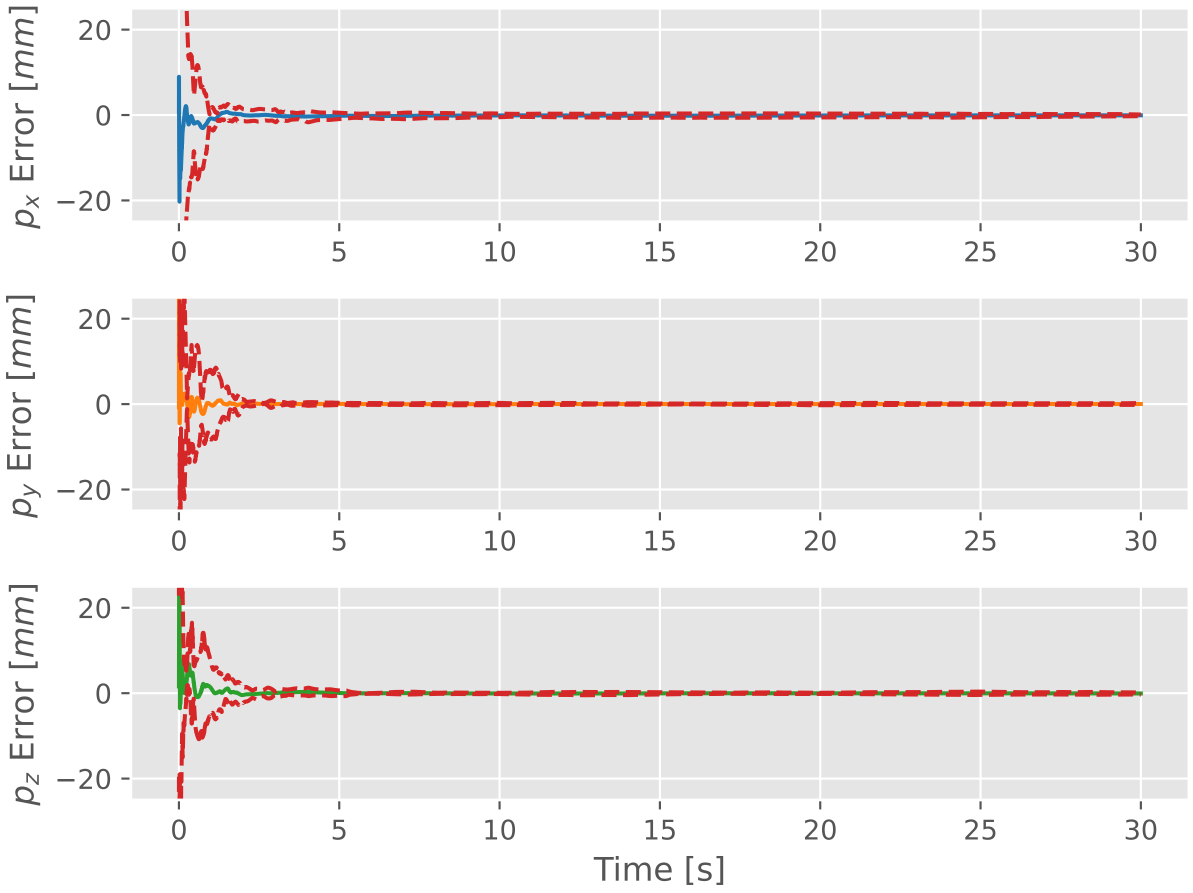
Experimental Testing

- Adjustable bar with multiple sensors
- Allows for rapid adjustments and runs
- Current Senors:
  - IMU:
    - Vectornav VN-100 & VN300
    - Pixhawk
    - DETA-10
  - Cameras:
    - Flir Blackfly
    - Basler Ace

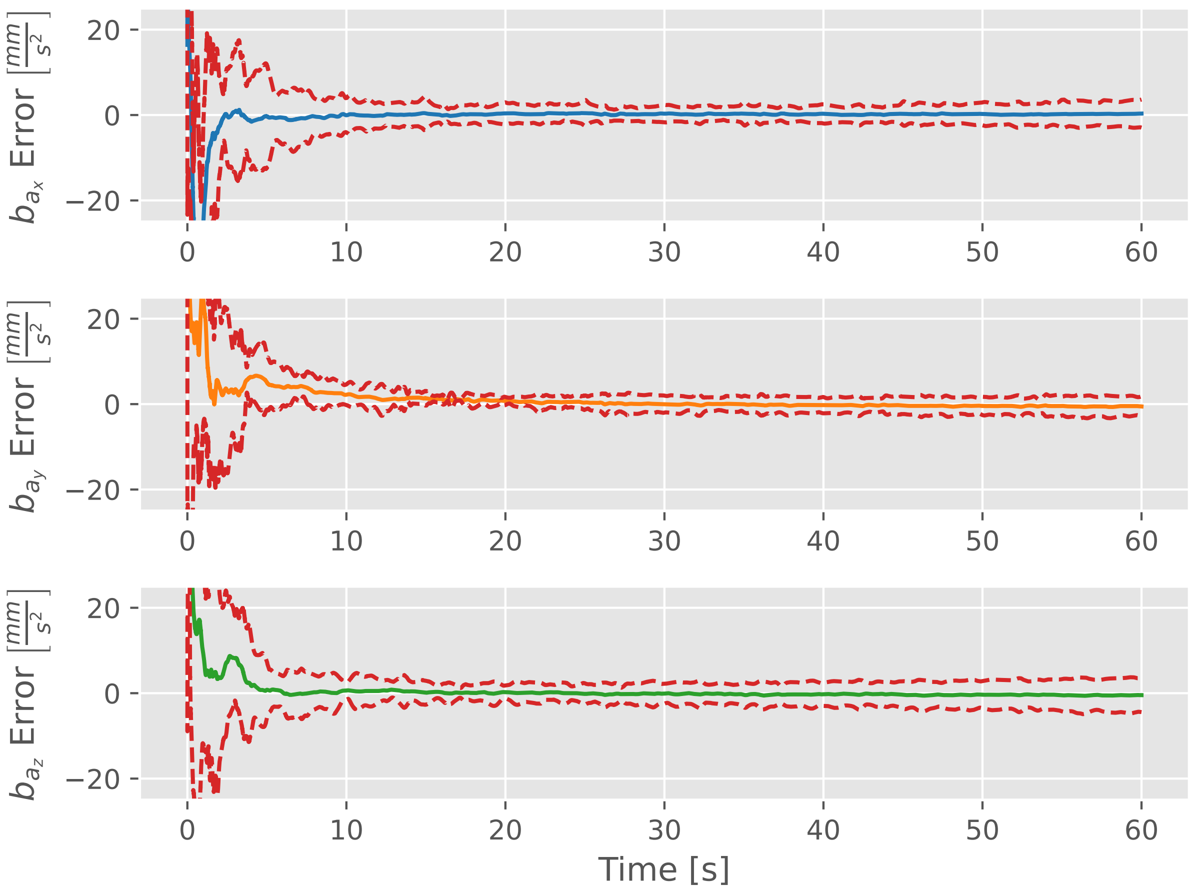


Experimental Results

Secondary IMU Position Convergence

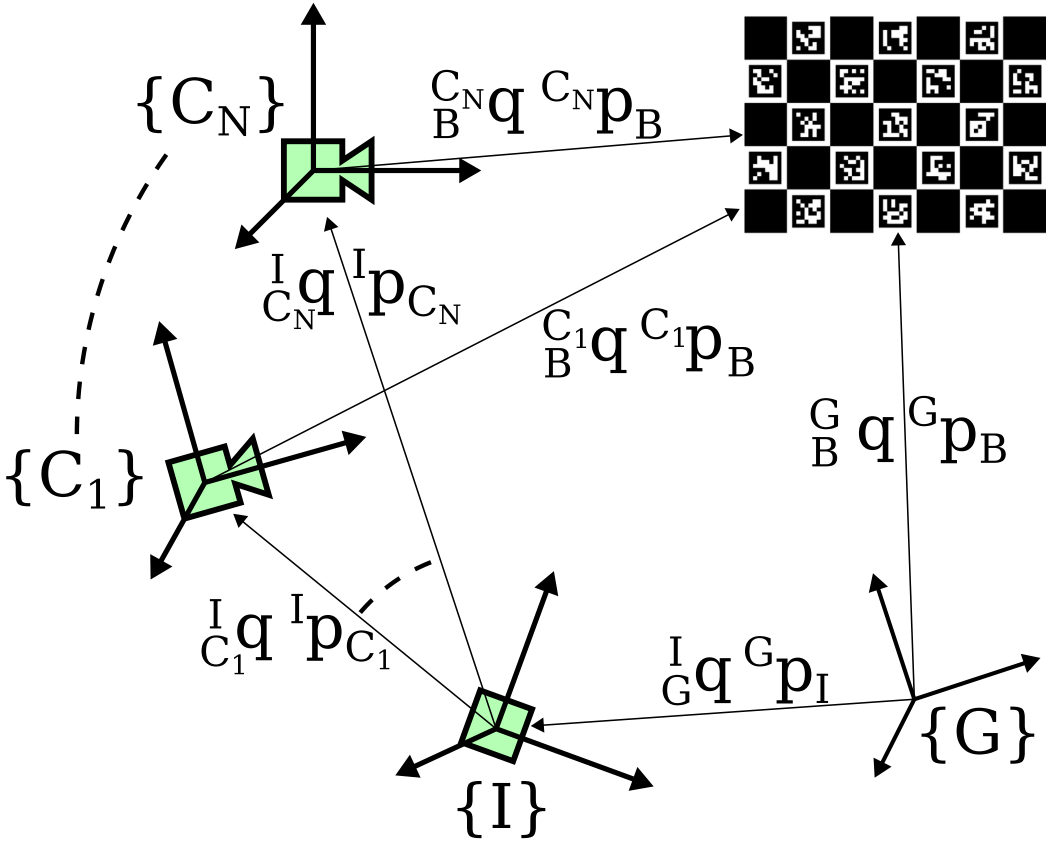


Secondary IMU Bias Convergence



Features in Development

- GPS Sensor Calibration
- Multiple Initialization Techniques
- Multi-Hypothesis Filtering
- Fiducial marker support



Takeaways

- Developed a multi-IMU calibration filter
- Utilized MSCKF framework for Visual Odometry
- Showed stability in simulation and experimentally
- Developed an open-sourced package for testing
- Try out EKF-CAL! We love feedback and collaboration!

Presentation References

1. J. Hartzer and S. Saripalli, " Online Multi Camera-IMU Calibration", IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2022. IEEE, arXiv

2. P. Jiang and S. Saripalli, "LiDARNet: A Boundary-Aware Domain Adaptation Model for Point Cloud Semantic Segmentation," 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 2021, pp. 2457-2464, IEEE

3. Experimental Evaluation of 3D-LIDAR Camera Extrinsic Calibration, S. Mishra, P. Osteen, G. Pandey and S. Saripalli, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS, 2020), arXiv

4. Extrinsic Calibration of a 3D-LIDAR and a Camera, S. Mishra, G. Pandey and S. Saripalli, IEEE Intelligent Vehicles Symposium (IV, 2020) arXiv

5. Chustz, G., & Saripalli, S. (2021). ROOAD: RELLIS Off-road Odometry Analysis Dataset. arXIv

Questions?

EKF-CAL Repository

Unmanned Systems Lab

TEXAS A&M UNIVERSITY

J. Mike Walker '66 Department of Mechanical Engineering