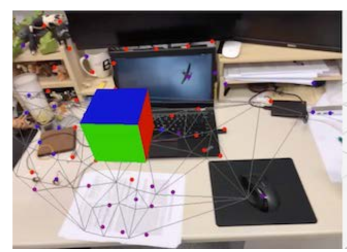
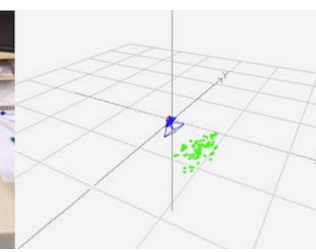
Monocular Visual-Inertial Fusion for State Estimation and Mapping

Shaojie Shen
Assistant Professor, HKUST

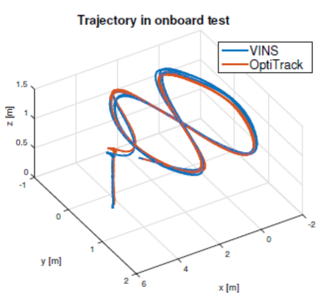
Why?

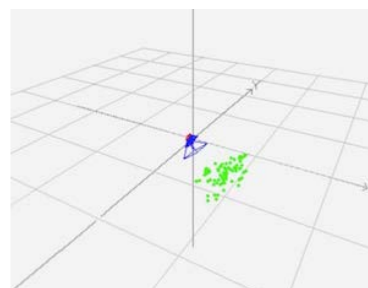
- State estimation for small drones that cannot afford stereo
- Mobile augmented reality

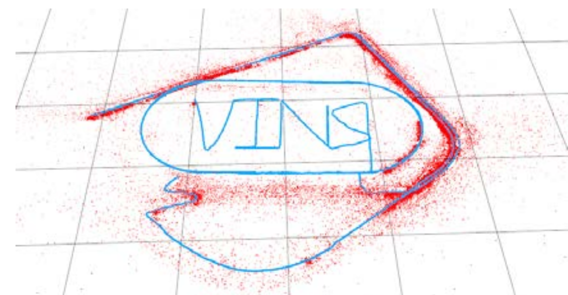
    

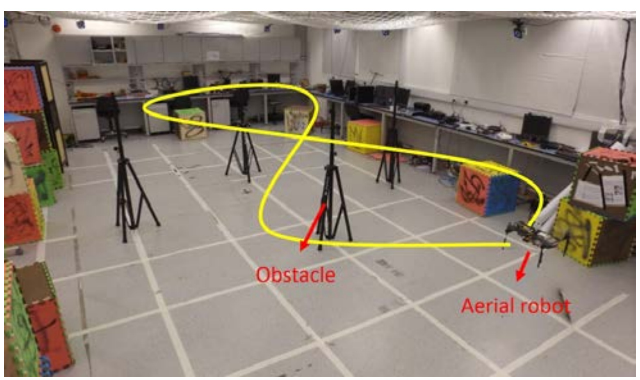
Requirements

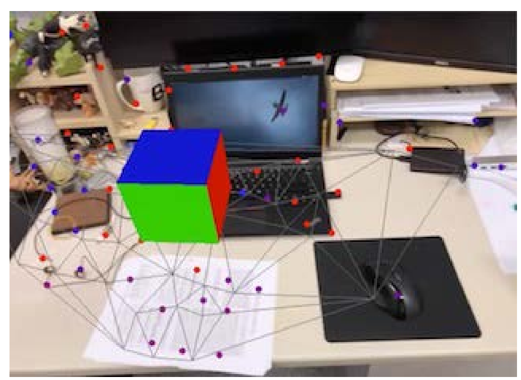
- Metric scale estimation
- Robust and smooth odometry – local accuracy
- Loop closure – global consistency









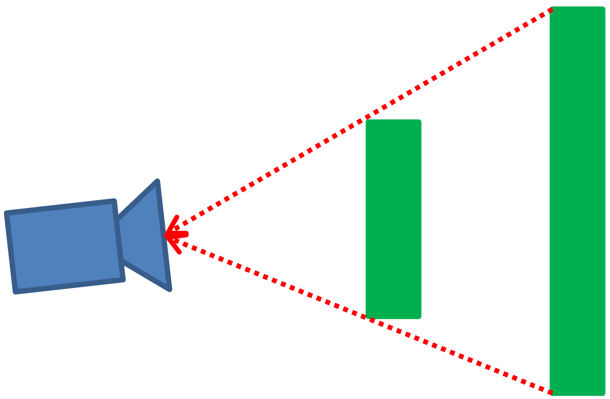


Related work

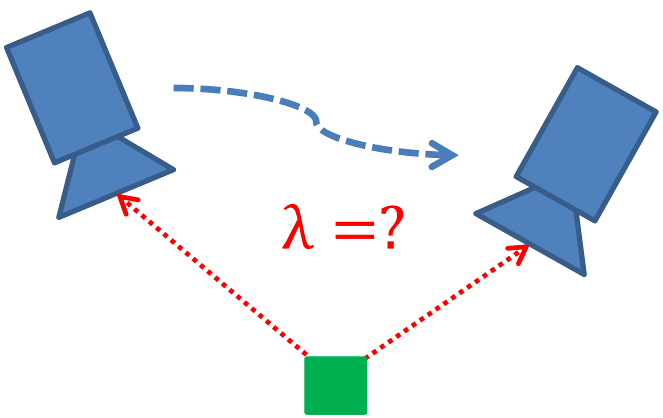
- MSC-KF (Mourikis and Roumeliotis, 2007)
  - Powers Google Tango
- okvis (Leutenegger, et al., 2015)
  - Code: https://github.com/ethz-asl/okvis
- Visual-Inertial ORB SLAM (Mur-Artal and Tardos, 2017)
  - No official source code available yet
- Apple ARKit
- Qualcomm Snapdragon 835

Challenges: Monocular Vision

- Scale ambiguity

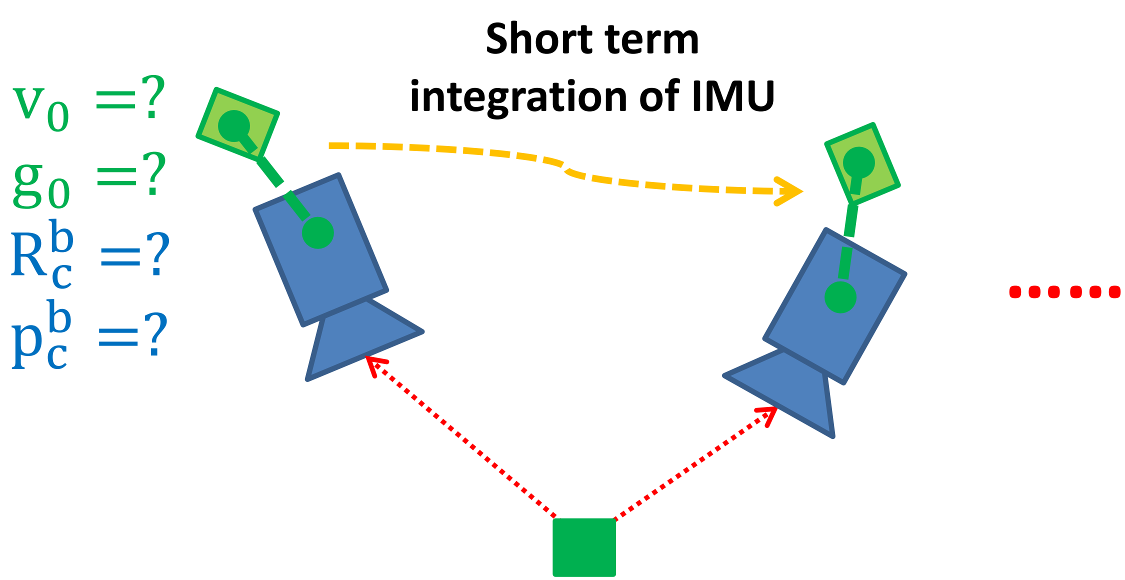


- Up-to-scale motion estimation and 3D reconstruction (Structure from Motion)



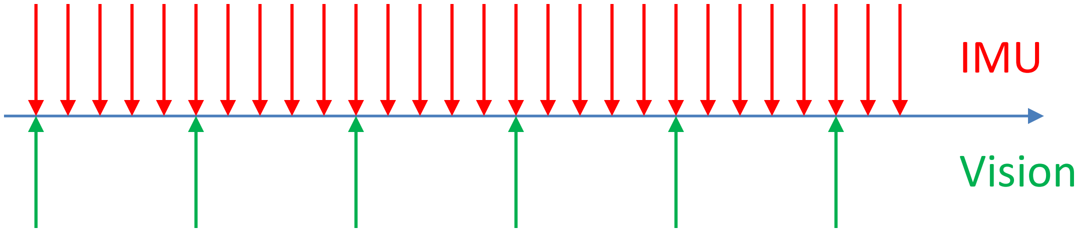
Challenges: Monocular Visual-Inertial Systems

- With IMU, scale is observable, but…
  - Requires recovery of initial velocity and attitude (gravity)
  - Requires online calibration camera-IMU extrinsic parameters
  - Requires multi-observation constraints

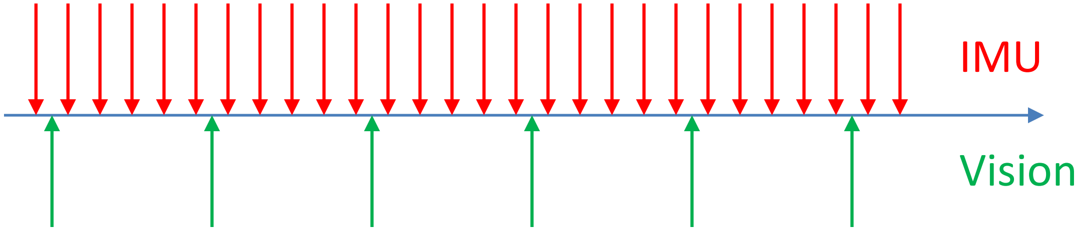


Challenges: Synchronization & Timestamps

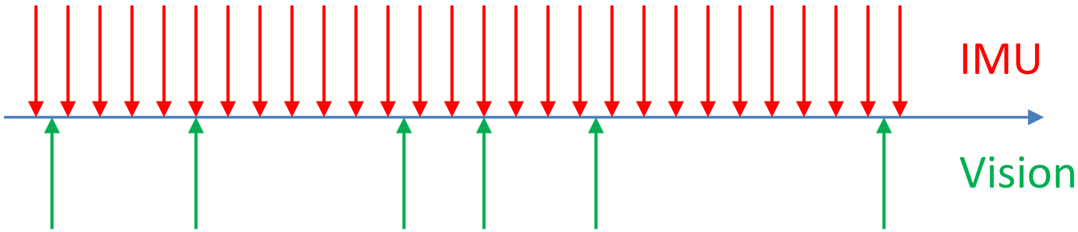
- Best: Sensors perfectly synchronized



- OK: Sensors have the same clock

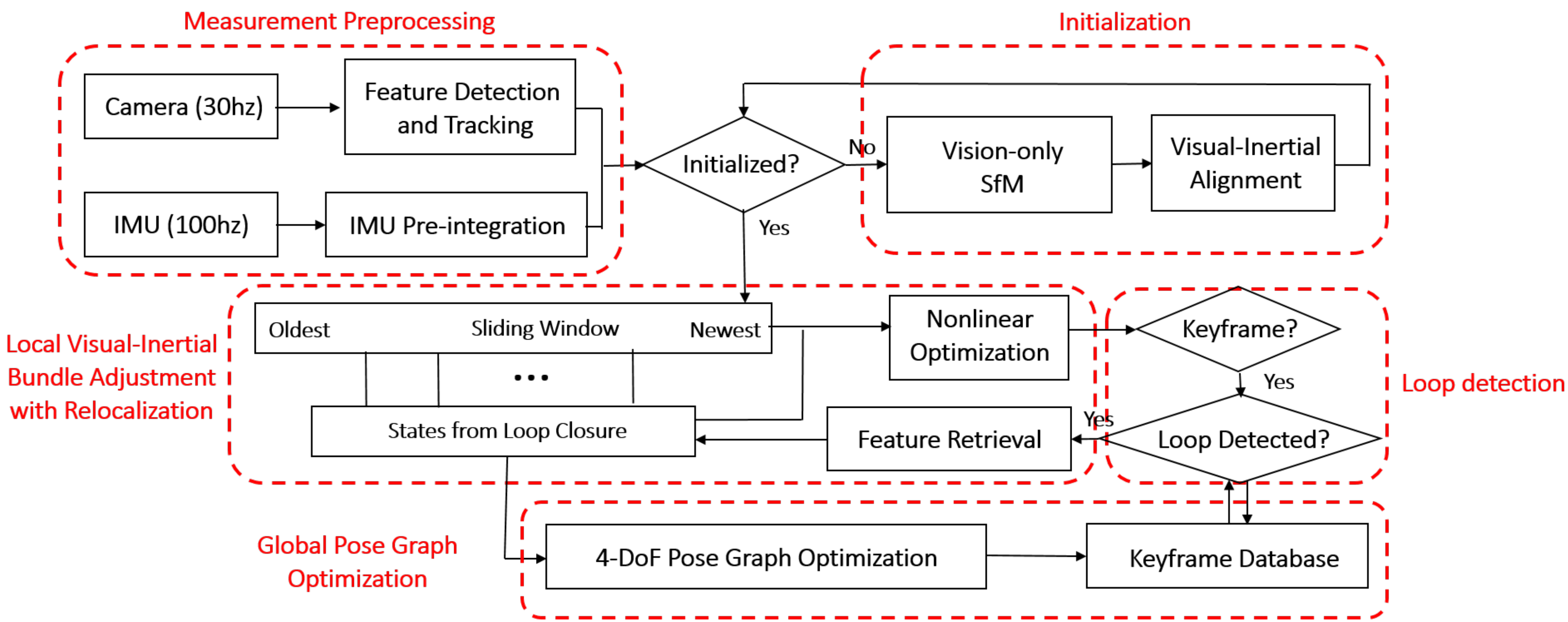


- Bad: Sensors with different clock or inaccurate timestamps



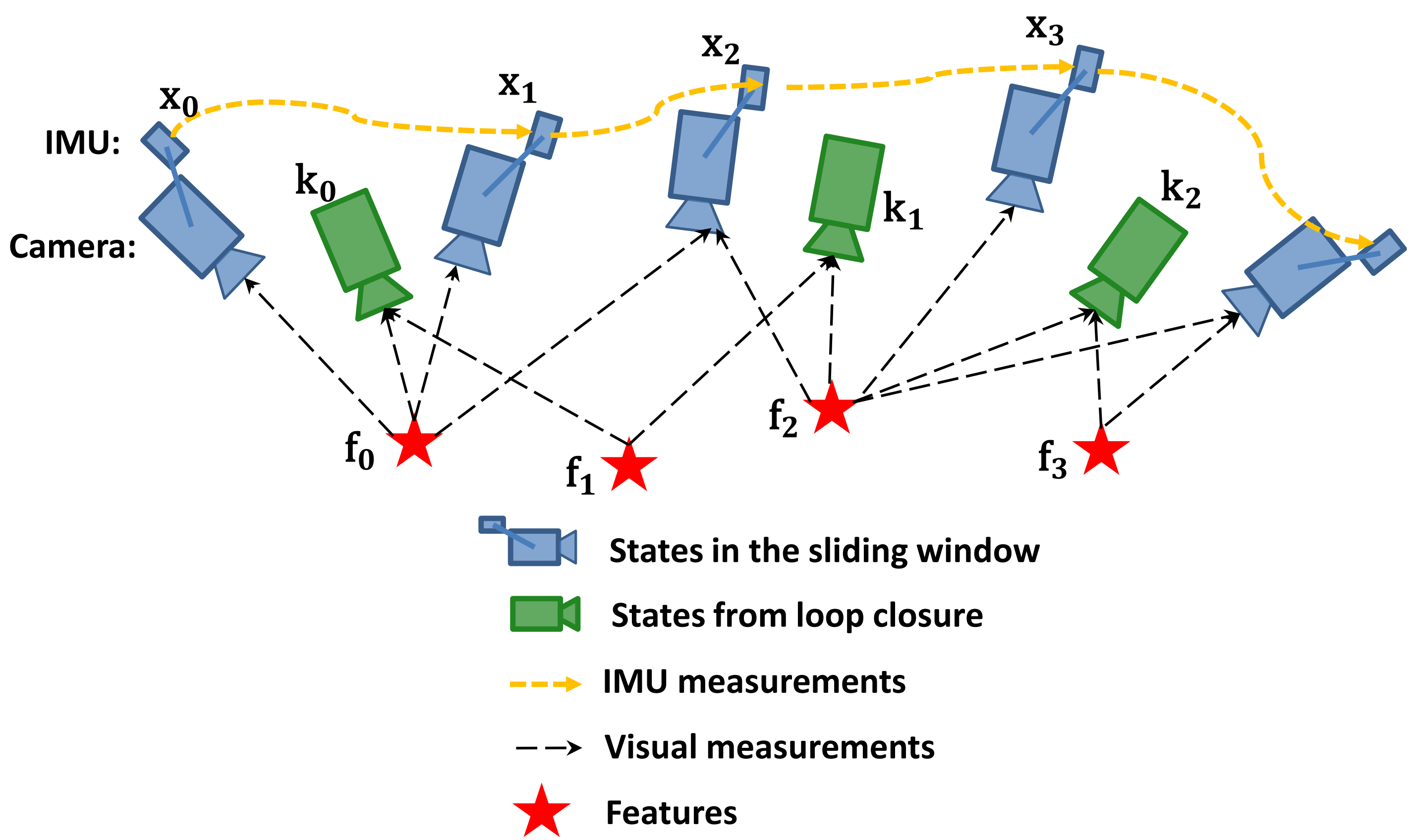
Monocular Visual-Inertial SLAM

- System diagram



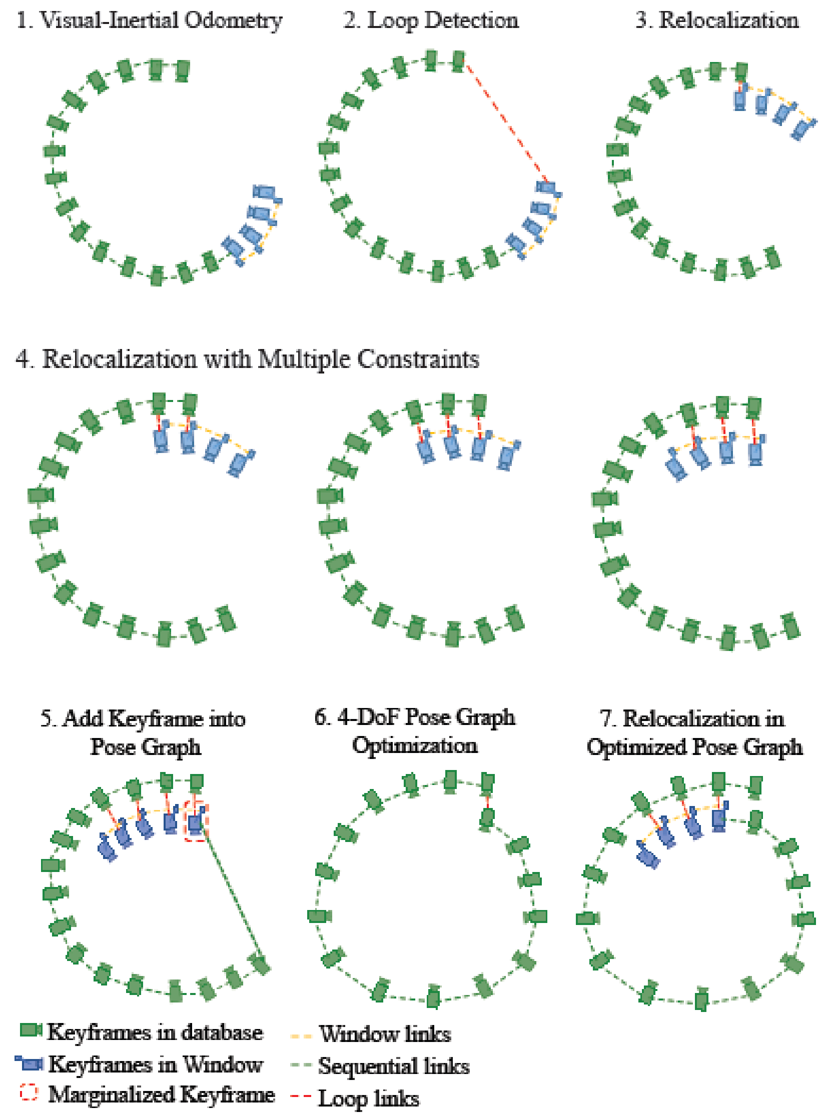
Monocular Visual-Inertial SLAM

- Monocular visual-inertial odometry with relocalization



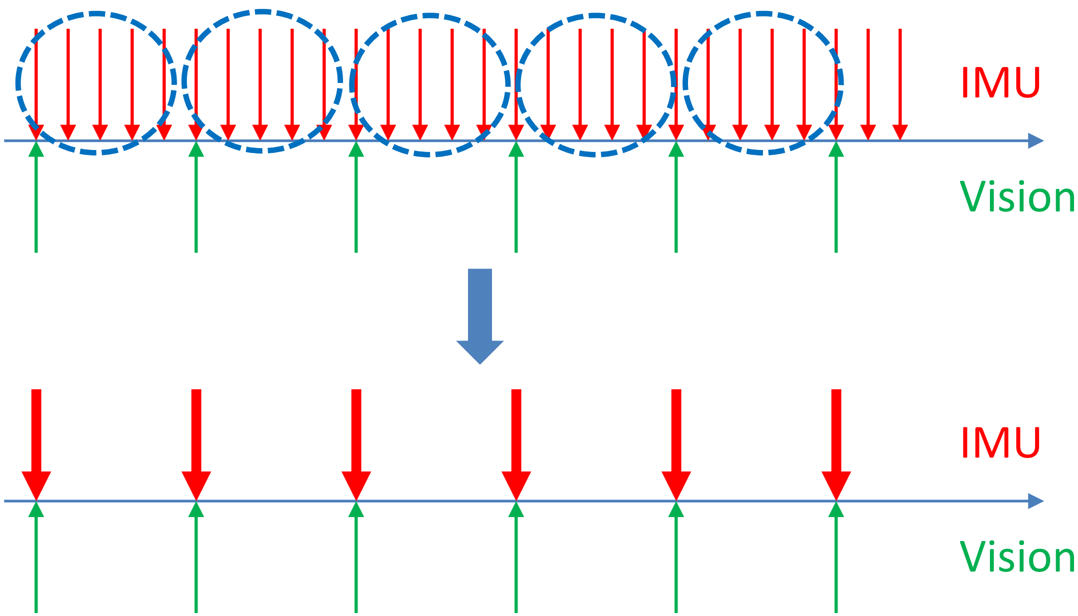
Monocular Visual-Inertial SLAM

- Global pose graph SLAM



How to Use IMU?

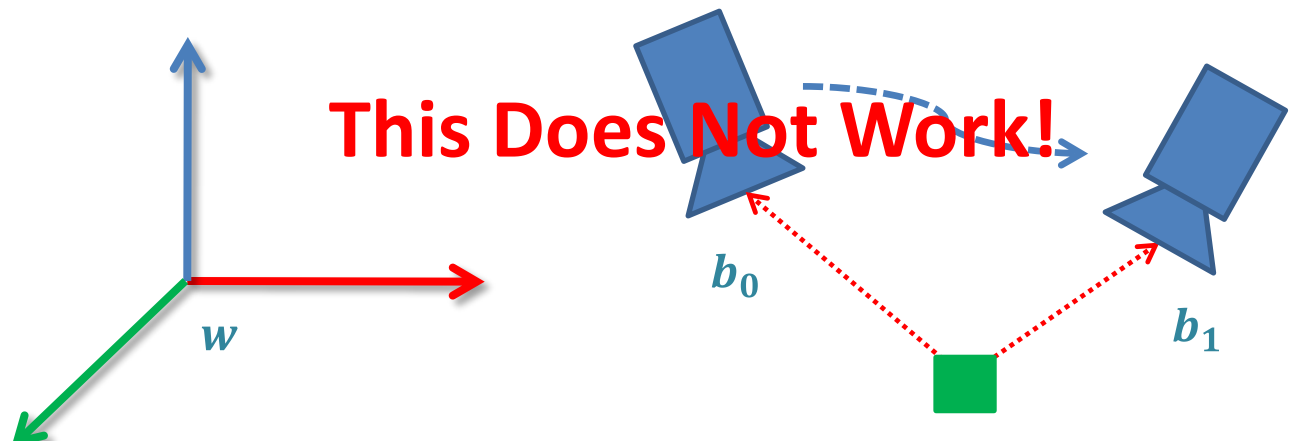
- IMU integration
  - IMU has higher rate than camera
  - Cannot estimate all IMU states
  - Need to integration IMU measurements



The Bad of IMU Integration in the Global Frame

- IMU integration in global frame
  - Requires global rotation at the time of integration

$$
\begin{aligned}
\mathbf{p}^w_{b_{k+1}} &= \mathbf{p}^w_{b_k} + \mathbf{v}^w_{b_k}\Delta t_k \\
&\quad + \iint_{t\in[k,k+1]} (\mathbf{R}^w_t(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t}) - \mathbf{g}^w)\, dt^2 \\
\mathbf{v}^w_{b_{k+1}} &= \mathbf{v}^w_{b_k} + \int_{t\in[k,k+1]} (\mathbf{R}^w_t(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t}) - \mathbf{g}^w)\, dt \\
\mathbf{q}^w_{b_{k+1}} &= \mathbf{q}^w_{b_k} \otimes \int_{t\in[k,k+1]} \mathbf{q}^w_t \otimes \begin{bmatrix} 0 \\ \frac{1}{2}(\hat{\boldsymbol{\omega}}_t - \mathbf{b}_{w_t}) \end{bmatrix} dt
\end{aligned}
$$

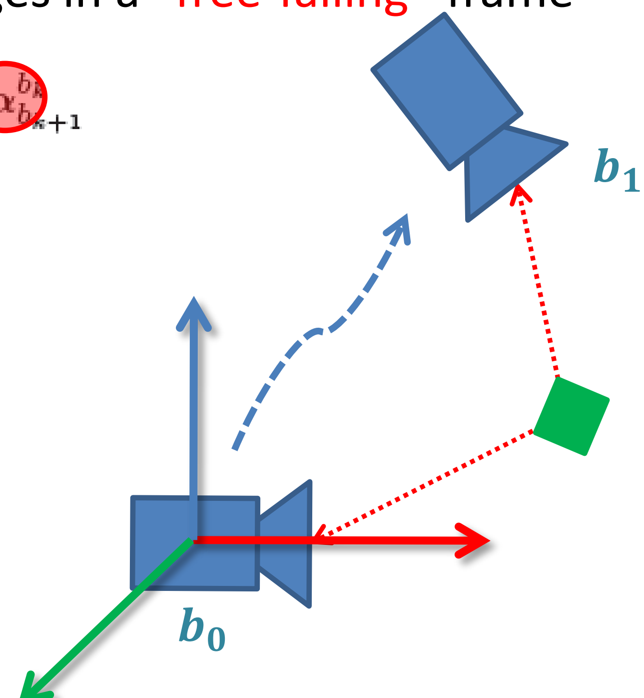


IMU Pre-Integration on Manifold

- IMU integration in the body frame of first pose of interests
  - IMU Integration without initialization
  - Can use any discrete implementation for numerical integration
  - Intuitive: "position" and "velocity" changes in a "free-falling" frame

$$
\begin{aligned}
\mathbf{R}_w^{b_k} \mathbf{p}_{b_{k+1}}^w &= \mathbf{R}_w^{b_k} \mathbf{p}_{b_k}^w + \mathbf{v}_{b_k}^w \Delta t_k - \frac{1}{2}\mathbf{g}^w \Delta t_k^2) + \alpha_{b_{k+1}}^{b_k} \\
\mathbf{R}_w^{b_k} \mathbf{v}_{b_{k+1}}^w &= \mathbf{R}_w^{b_k} \mathbf{v}_{b_k}^w + \beta_{b_{k+1}}^{b_k} \\
\mathbf{q}_w^{b_k} \otimes \mathbf{q}_{b_{k+1}}^w &= \gamma_{b_{k+1}}^{b_k}
\end{aligned}
$$

$$
\begin{aligned}
\alpha_{b_{k+1}}^{b_k} &= \iint_{t\in[k,k+1]} \mathbf{R}_{b_t}^{b_k} (\hat{\mathbf{a}}_t - \mathbf{b}_{a_t}) dt^2 \\
\beta_{b_{k+1}}^{b_k} &= \int_{t\in[k,k+1]} \mathbf{R}_{b_t}^{b_k} (\hat{\mathbf{a}}_t - \mathbf{b}_{a_t}) dt \\
\gamma_{b_{k+1}}^{b_k} &= \int_{t\in[k,k+1]} \gamma_{b_t}^{b_k} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}(\hat{\omega}_t - \mathbf{b}_{w_t}) \end{bmatrix} dt,
\end{aligned}
$$



IMU Pre-Integration on Manifold

- Uncertainty propagation on manifold
  - Derive the error state model for the IMU pre-integration dynamics

$$\begin{bmatrix} \delta\dot{\boldsymbol{\alpha}}_t^{b_k} \\ \delta\dot{\boldsymbol{\beta}}_t^{b_k} \\ \delta\dot{\boldsymbol{\theta}}_t^{b_k} \\ \delta\dot{\mathbf{b}}_{a_t} \\ \delta\dot{\mathbf{b}}_{w_t} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{I} & 0 & 0 & 0 \\ 0 & 0 & -\mathbf{R}_t^{b_k}\lfloor\hat{\mathbf{a}}_t - \mathbf{b}_{a_t}\rfloor_\times & -\mathbf{R}_t^{b_k} & 0 \\ 0 & 0 & -\lfloor\hat{\boldsymbol{\omega}}_t - \mathbf{b}_{w_t}\rfloor_\times & 0 & -\mathbf{I} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\alpha}_t^{b_k} \\ \delta\boldsymbol{\beta}_t^{b_k} \\ \delta\boldsymbol{\theta}_t^{b_k} \\ \delta\mathbf{b}_{a_t} \\ \delta\mathbf{b}_{w_t} \end{bmatrix}$$

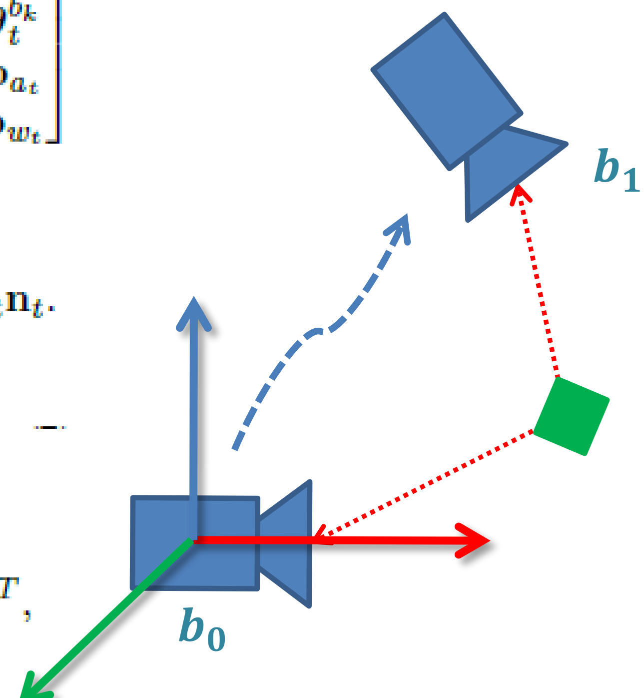
Bias uncertainty

$$+ \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\mathbf{R}_t^{b_k} & 0 & 0 & 0 \\ 0 & -\mathbf{I} & 0 & 0 \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{n}_a \\ \mathbf{n}_w \\ \mathbf{n}_{b_a} \\ \mathbf{n}_{b_w} \end{bmatrix} = \mathbf{F}_t \delta\mathbf{z}_t^{b_k} + \mathbf{G}_t \mathbf{n}_t.$$

  - Discrete-time implementation

$$\mathbf{P}_{t+\delta t}^{b_k} = (\mathbf{I} + \mathbf{F}_t\delta t)\mathbf{P}_t^{b_k}(\mathbf{I} + \mathbf{F}_t\delta t)^T + (\mathbf{G}_t\delta t)\mathbf{Q}(\mathbf{G}_t\delta t)^T,$$
$$t \in [k, k+1],$$

Covariance matrix for pre-integrated IMU measurements



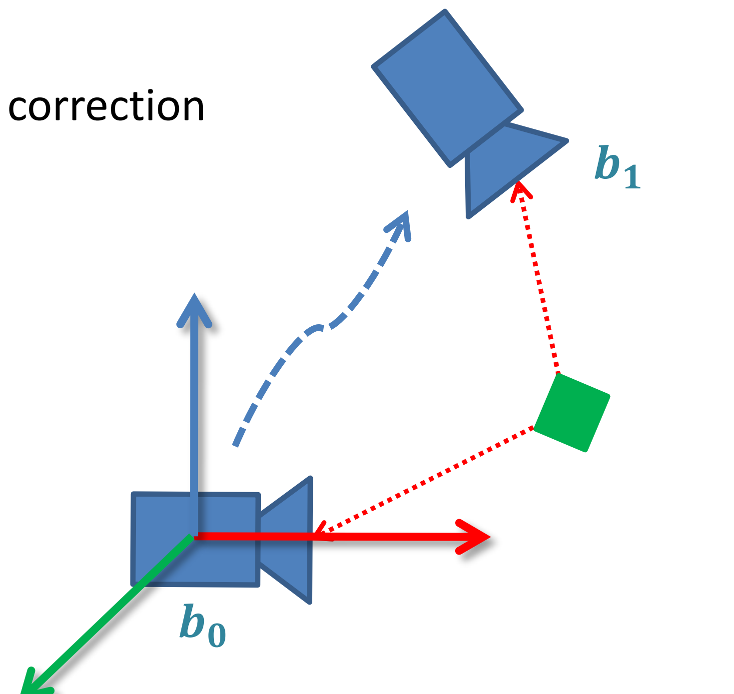
IMU Pre-Integration on Manifold

- Jacobian matrices for bias correction
  - Also derive the Jacobian of the pre-integrated measurements w.r.t. IMU bias

$$
\begin{aligned}
\mathbf{J}_{b_k} &= \mathbf{I}, \\
\mathbf{J}_{t+\delta t} &= (\mathbf{I} + \mathbf{F}_t \delta t)\mathbf{J}_t, \quad t \in [k, k+1]
\end{aligned}
$$

  - And write down the linearized model for bias correction

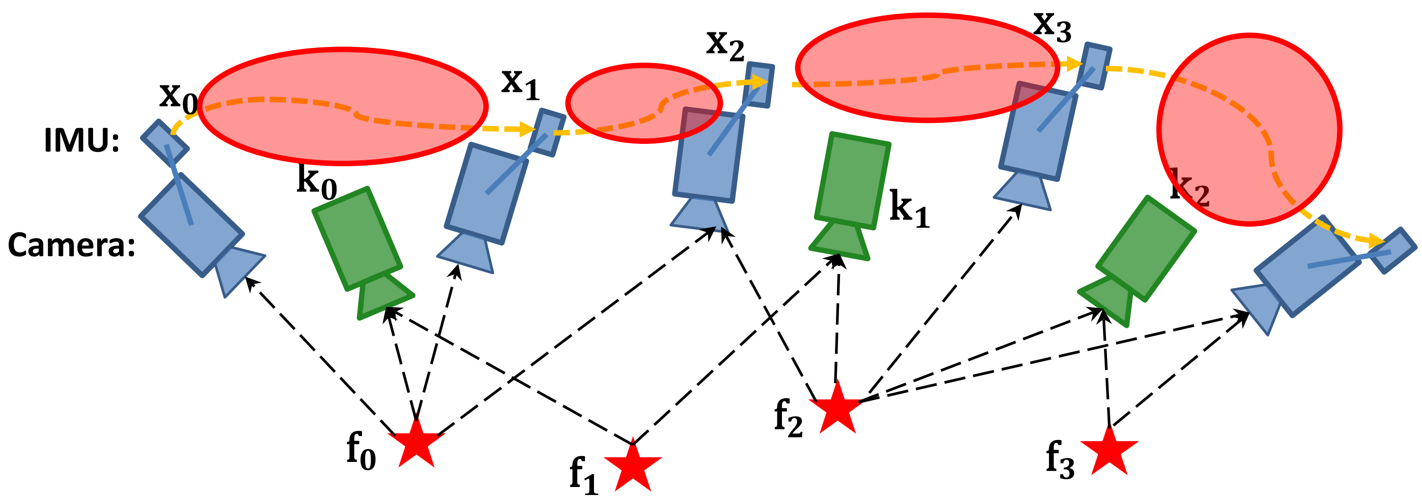
$$
\begin{aligned}
\alpha^{b_k}_{b_{k+1}} &\approx \hat{\alpha}^{b_k}_{b_{k+1}} + \mathbf{J}^{\alpha}_{b_a} \delta \mathbf{b}_{a_k} + \mathbf{J}^{\alpha}_{b_w} \delta \mathbf{b}_{w_k} \\
\beta^{b_k}_{b_{k+1}} &\approx \hat{\beta}^{b_k}_{b_{k+1}} + \mathbf{J}^{\beta}_{b_a} \delta \mathbf{b}_{a_k} + \mathbf{J}^{\beta}_{b_w} \delta \mathbf{b}_{w_k} \\
\gamma^{b_k}_{b_{k+1}} &\approx \hat{\gamma}^{b_k}_{b_{k+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}^{\gamma}_{b_w} \delta \mathbf{b}_{w_k} \end{bmatrix}
\end{aligned}
$$



IMU Pre-Integration on Manifold

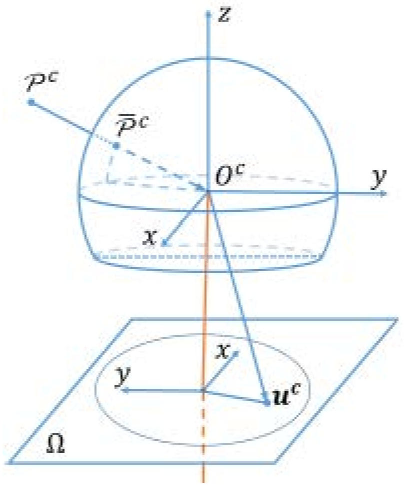
- Pre-integrated IMU measurement model
  - Describes the spatial and uncertainty relations between two states in the local sliding window

$$\begin{bmatrix} \hat{\alpha}^{b_k}_{b_{k+1}} \\ \hat{\beta}^{b_k}_{b_{k+1}} \\ \hat{\gamma}^{b_k}_{b_{k+1}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{R}^{b_k}_{w}(\mathbf{p}^w_{b_{k+1}} - \mathbf{p}^w_{b_k} + \frac{1}{2}\mathbf{g}^w \Delta t_k^2 - \mathbf{v}^w_{b_k}\Delta t_k) \\ \mathbf{R}^{b_k}_{w}(\mathbf{v}^w_{b_{k+1}} + \mathbf{g}^w \Delta t_k - \mathbf{v}^w_{b_k}) \\ {\mathbf{q}^w_{b_k}}^{-1} \otimes \mathbf{q}^w_{b_{k+1}} \\ \mathbf{b}_{ab_{k+1}} - \mathbf{b}_{ab_k} \\ \mathbf{b}_{wb_{k+1}} - \mathbf{b}_{wb_k} \end{bmatrix}$$



Vision Front-End

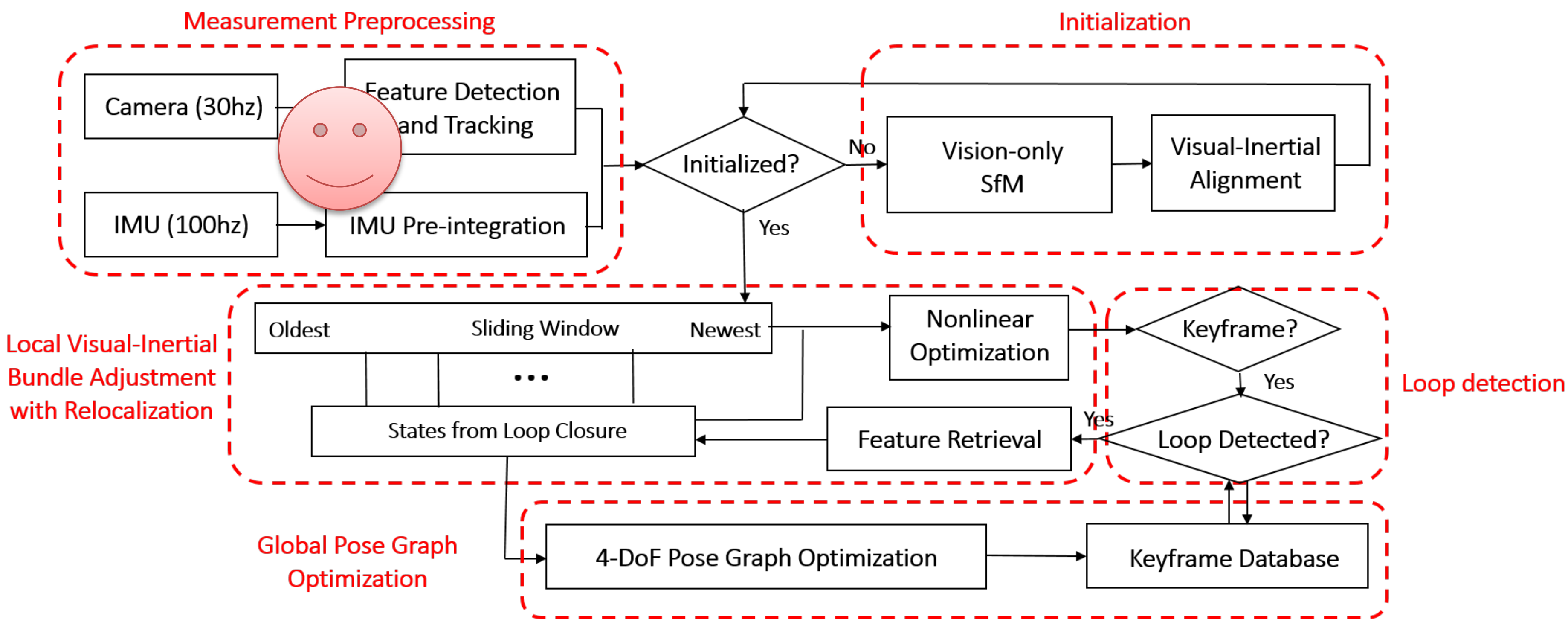
- Feature processing pipeline
  - Harris corners…
  - KLT tracker...
  - Track between consecutive frames
  - RANSAC for preliminary outlier removal
  - Unified camera model for fisheye cameras



- Keyframe selection
  - Case 1: Rotation-compensated average feature parallax++
  - Case 2: Number of tracked features--

Quick Review

- System diagram

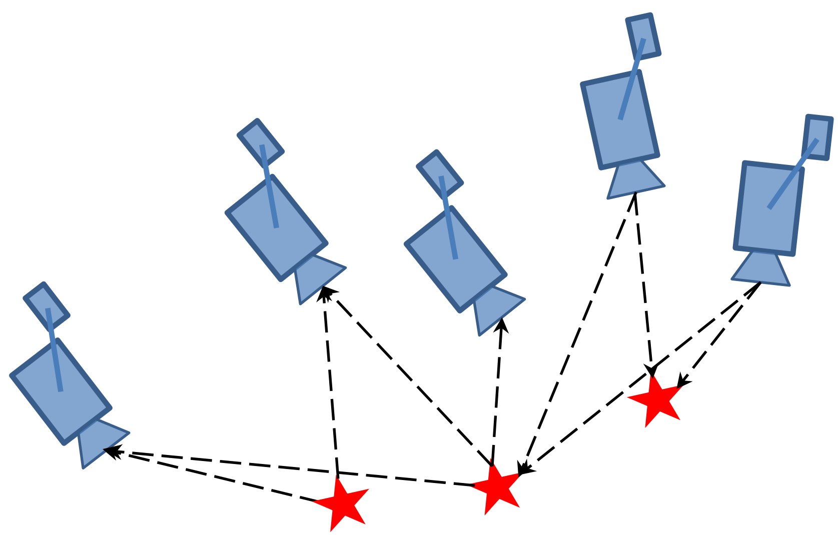


Estimator Initialization

- Very, very, very important for monocular visual-inertial systems
- Pipeline
  - Monocular vision-only SFM in a local window
  - Camera-IMU rotation calibration
  - Visual-inertial alignment

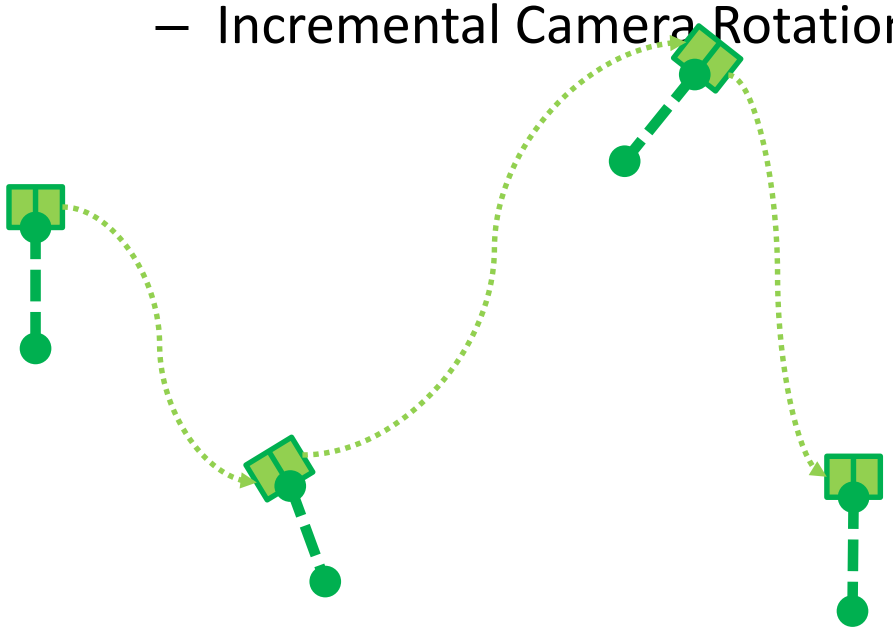
Estimator Initialization

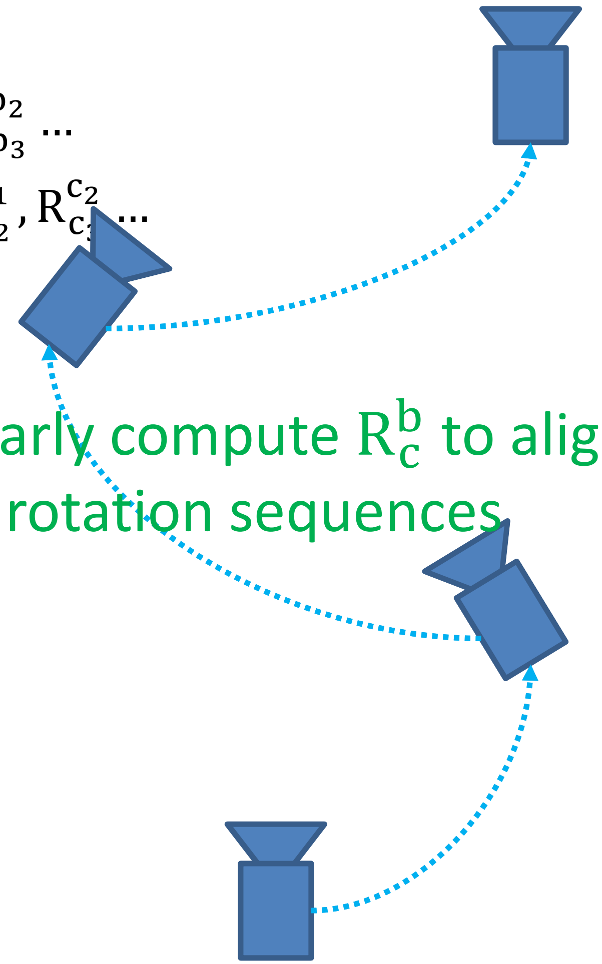
- Monocular vision-only SFM
  - In a small window (10 frames, 1sec)
  - Up-to-scale, locally drift-free position estimates
  - Not aligned with gravity



Estimator Initialization

- Camera-IMU rotation calibration
  - Incremental IMU Rotations: $R_{b_1}^{b_0}, R_{b_2}^{b_1}, R_{b_3}^{b_2}$ …
  - Incremental Camera Rotations: $R_{c_1}^{c_0}, R_{c_2}^{c_1}, R_{c_3}^{c_2}$ …



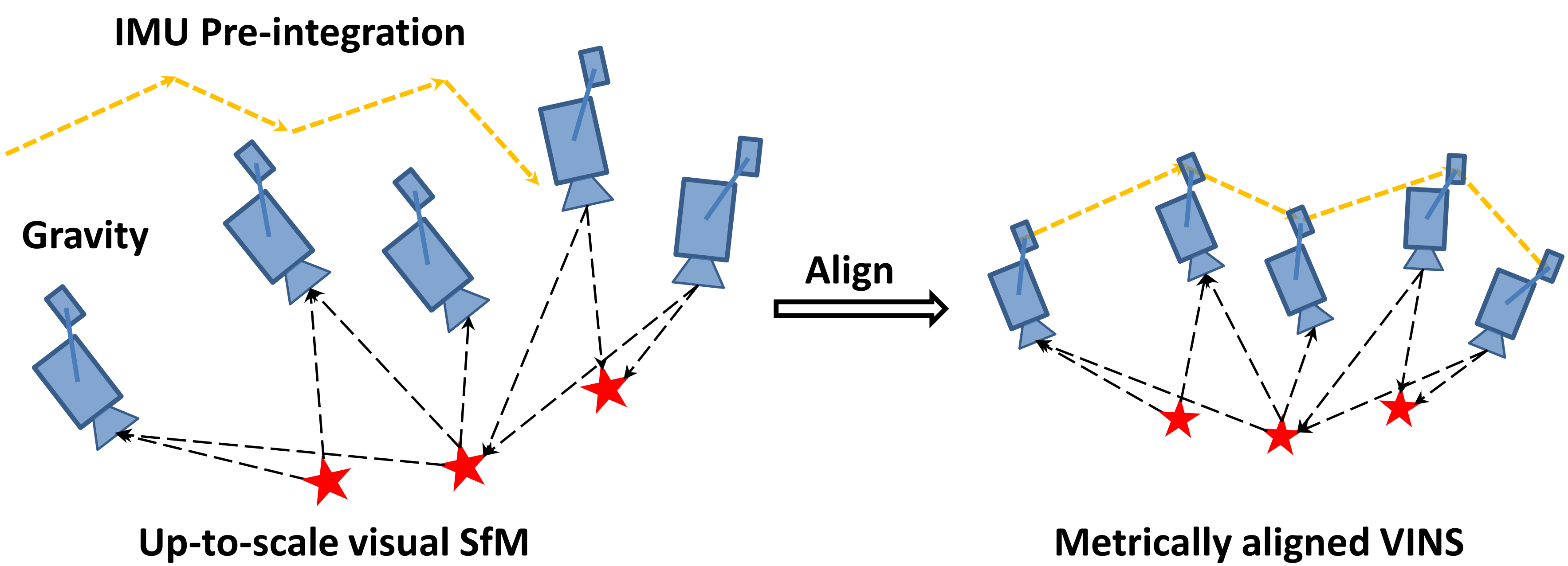


Linearly compute R_c^b to align two rotation sequences

Estimator Initialization

- Visual-inertial alignment
  - Estimates velocity of each frame, gravity vector, and scale

$$\mathcal{X}_I = \left[\mathbf{v}_{b_0}^{c_0}, \mathbf{v}_{b_1}^{c_0}, \cdots \mathbf{v}_{b_n}^{c_0}, \mathbf{g}^{c_0}, s\right]$$



Estimator Initialization

- Visual-inertial alignment
  - Linear measurement model

IMU Pre-integration

Known values from vSFM

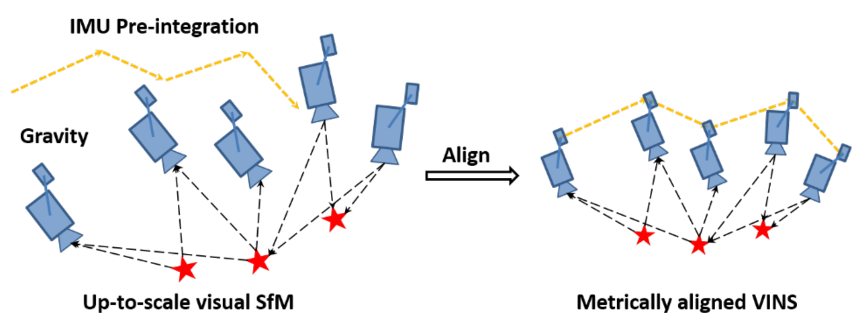
$$\hat{\mathbf{z}}^{b_k}_{b_{k+1}} = \begin{bmatrix} \hat{\alpha}^{b_k}_{b_{k+1}} - \mathbf{R}^{c_0}_{c_{k+1}} \mathbf{p}^c_b + \mathbf{R}^{c_0}_{c_k} \mathbf{p}^c_b \\ \hat{\beta}^{b_k}_{b_{k+1}} \end{bmatrix} = \mathbf{H}^{b_k}_{b_{k+1}} \mathcal{X}_I + \mathbf{n}^{b_k}_{b_{k+1}}$$

$$\approx \begin{bmatrix} -\mathbf{R}^{b_k}_{c_0} \Delta t_k & \mathbf{0} & \frac{1}{2}\mathbf{R}^{b_k}_{c_0} \Delta t_k^2 & \mathbf{R}^{b_k}_{c_0}(\bar{\mathbf{p}}^{c_0}_{c_{k+1}} - \bar{\mathbf{p}}^{c_0}_{c_k}) \\ -\mathbf{R}^{b_k}_{c_0} & \mathbf{R}^{b_k}_{c_0} & \mathbf{R}^{b_k}_{c_0} \Delta t_k & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}^{c_0}_{b_k} \\ \mathbf{v}^{c_0}_{b_{k+1}} \\ \mathbf{g}^{c_0} \\ s \end{bmatrix}$$

States to be initialized

  - Solve a linear system
    - Scale and rotate the vSFM

$$\min_{\mathcal{X}_I} \sum_{k \in \mathcal{B}} \left\| \hat{\mathbf{z}}^{b_k}_{b_{k+1}} - \mathbf{H}^{b_k}_{b_{k+1}} \mathcal{X}_I \right\|^2$$

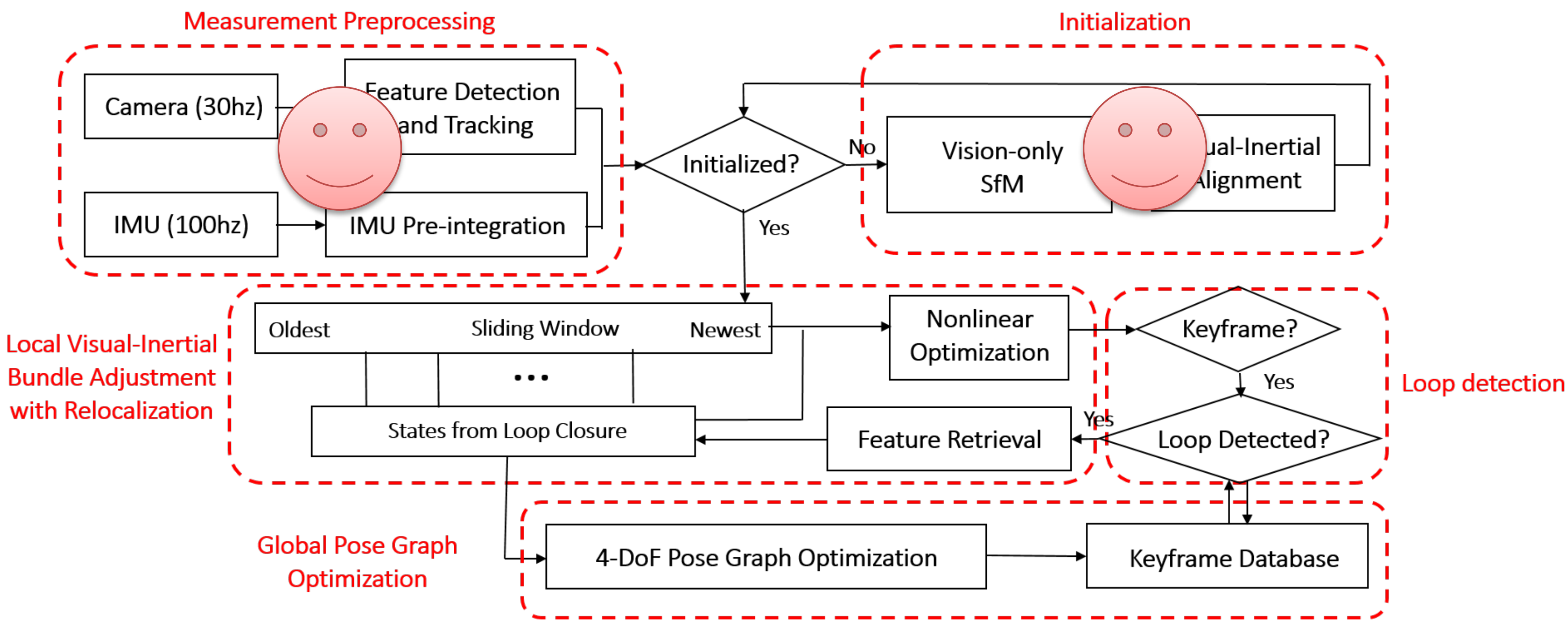


Estimator Initialization

- Current issues:
  - IMU biases are not initialized
    - Gyroscope: obtained from stationary measurements
    - Accelerometer: problematic…
  - May fail at high altitude scenes due to excessive IMU integration time
    - Solution: Spline-based initialization, use derivatives instead of integration
    - T. Liu and S. Shen. High altitude monocular visual-inertial state estimation: initialization and sensor fusion. In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), Singapore, May 2017

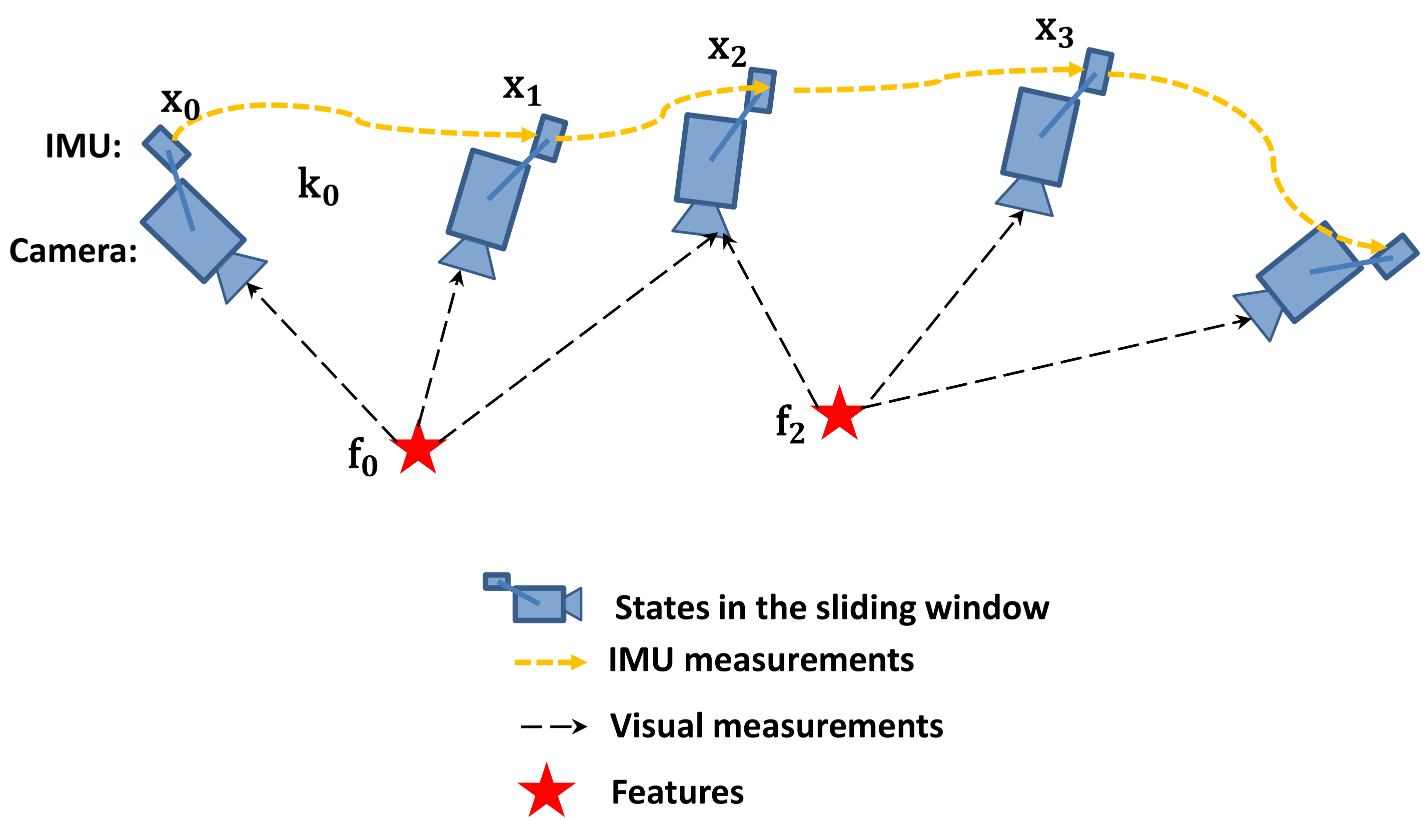
Quick Review

- System diagram



Monocular Visual-Inertial Odometry

- Nonlinear graph optimization-based, tightly-coupled, sliding window, visual-inertial bundle adjustment



Monocular Visual-Inertial Odometry

- Nonlinear graph-based optimization
  - Optimize position, velocity, rotation, IMU biases, inverse feature depth, and camera-IMU transformation simultaneously:

$$
\begin{aligned}
\mathcal{X} &= \left[\mathbf{x}_0,\, \mathbf{x}_1,\, \cdots\, \mathbf{x}_n,\, \mathbf{x}_c^b,\, \lambda_0,\, \lambda_1,\, \cdots\, \lambda_m\right] \\
\mathbf{x}_k &= \left[\mathbf{p}_{b_k}^w,\, \mathbf{v}_{b_k}^w,\, \mathbf{q}_{b_k}^w,\, \mathbf{b}_a,\, \mathbf{b}_g\right],\, k \in [0, n] \\
\mathbf{x}_c^b &= \left[\mathbf{p}_c^b,\, \mathbf{q}_c^b\right],
\end{aligned}
$$

  - Minimize residuals from all sensors

IMU measurement residual

$$
\min_{\mathcal{X}} \left\{ \left\| \mathbf{r}_p - \mathbf{H}_p \mathcal{X} \right\|^2 + \sum_{k \in \mathcal{B}} \left\| \mathbf{r}_{\mathcal{B}}\left(\hat{\mathbf{z}}_{b_{k+1}}^{b_k}, \mathcal{X}\right) \right\|_{\mathbf{P}_{b_{k+1}}^{b_k}}^2 + \sum_{(l,j) \in \mathcal{C}} \left\| \mathbf{r}_{\mathcal{C}}\left(\hat{\mathbf{z}}_l^{c_j}, \mathcal{X}\right) \right\|_{\mathbf{P}_l^{c_j}}^2 \right\}
$$

Prior from marginalization

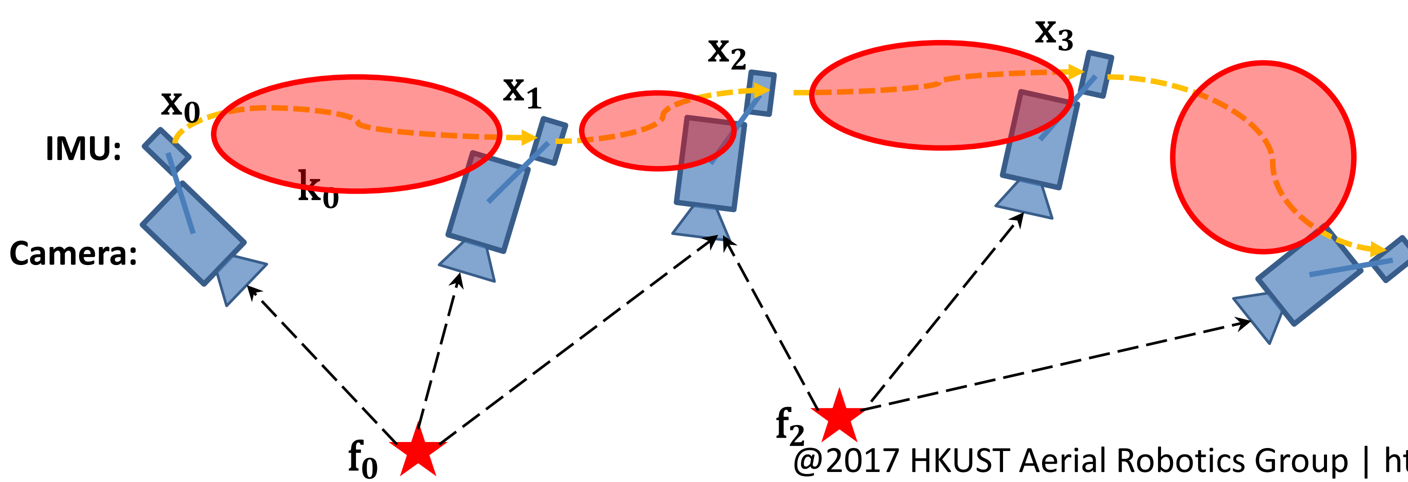
Vision measurement residual

Monocular Visual-Inertial Odometry

- IMU measurement residual
  - Additive for "position" and "velocity" changes, and biases
  - Multiplicative for incremental rotation

IMU pre-integration "blocks"

$$\mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}^{b_k}_{b_{k+1}}, \mathcal{X}) = \begin{bmatrix} \delta\alpha^{b_k}_{b_{k+1}} \\ \delta\beta^{b_k}_{b_{k+1}} \\ \delta\theta^{b_k}_{b_{k+1}} \\ \delta\mathbf{b}_a \\ \delta\mathbf{b}_g \end{bmatrix} = \begin{bmatrix} \mathbf{R}^{b_k}_w(\mathbf{p}^w_{b_{k+1}} - \mathbf{p}^w_{b_k} + \frac{1}{2}\mathbf{g}^w\Delta t_k^2 - \mathbf{v}^w_{b_k}\Delta t_k) - \hat{\alpha}^{b_k}_{b_{k+1}} \\ \mathbf{R}^{b_k}_w(\mathbf{v}^w_{b_{k+1}} + \mathbf{g}^w\Delta t_k - \mathbf{v}^w_{b_k}) - \hat{\beta}^{b_k}_{b_{k+1}} \\ 2\left[\mathbf{q}^{w^{-1}}_{b_{k+1}} \otimes \mathbf{q}^w_{b_k} \otimes \hat{\gamma}^{b_k}_{b_{k+1}}\right]_{xyz} \\ \mathbf{b}_{ab_{k+1}} - \mathbf{b}_{ab_k} \\ \mathbf{b}_{wb_{k+1}} - \mathbf{b}_{wb_k} \end{bmatrix}$$

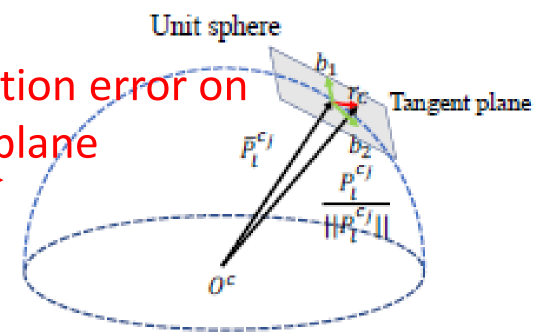


Monocular Visual-Inertial Odometry

- Vision measurement residual
  - Spherical camera model
  - At least 2 observations per feature

$$\mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}_l^{c_j}, \mathcal{X}) = [\mathbf{b}_1 \ \ \mathbf{b}_2]^T \cdot (\bar{\mathcal{P}}_l^{c_j} - \frac{\mathcal{P}_l^{c_j}}{\|\mathcal{P}_l^{c_j}\|})$$

Reprojection error on tangent plane

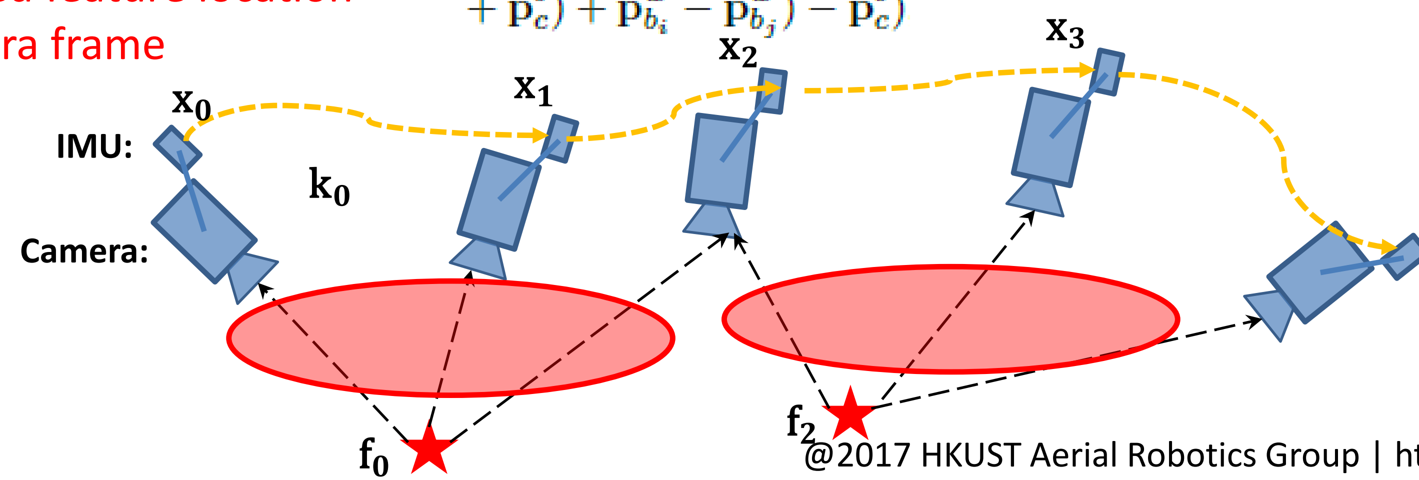


$$\bar{\mathcal{P}}_l^{c_j} = \pi_c^{-1}(\begin{bmatrix}\hat{u}_l^{c_j}\\ \hat{v}_l^{c_j}\end{bmatrix})$$

Estimated feature location from 2D measurements and depth

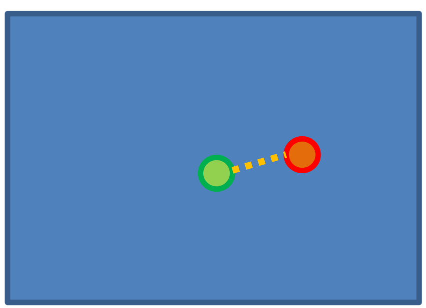
$$\mathcal{P}_l^{c_j} = \mathbf{R}_b^c(\mathbf{R}_w^{b_j}(\mathbf{R}_{b_i}^w(\mathbf{R}_c^b \frac{1}{\lambda_l}\pi_c^{-1}(\begin{bmatrix}u_l^{c_i}\\ v_l^{c_i}\end{bmatrix}) + \mathbf{p}_c^b) + \mathbf{p}_{b_i}^w - \mathbf{p}_{b_j}^w) - \mathbf{p}_c^b)$$

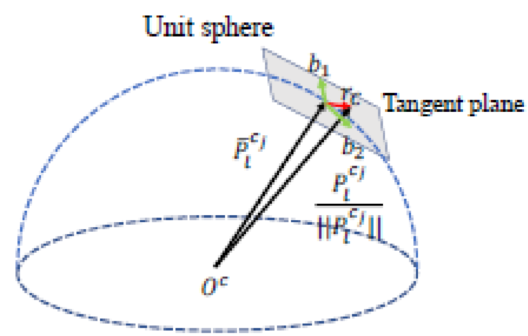
Predicted feature location in camera frame



Monocular Visual-Inertial Odometry

- Spherical vs. pinhole camera models
  - Different ways to define the reprojection error
  - Able to model cameras with arbitrary FOV





Spherical

Monocular Visual-Inertial Odometry

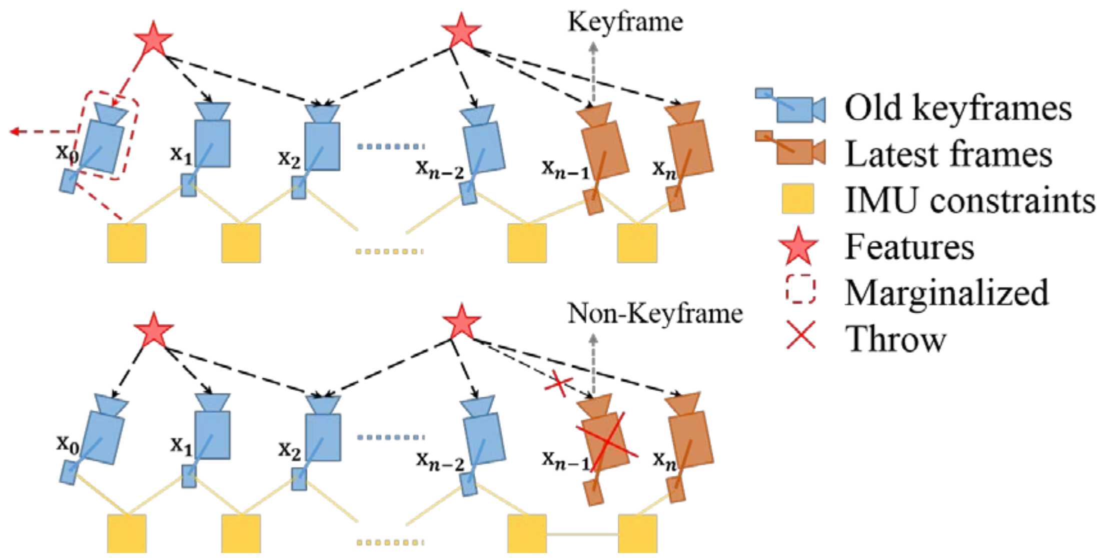
- Solving the nonlinear system
  - Minimize residuals from all sensors

$$\min_{\mathcal{X}} \left\{ \left\| \mathbf{r}_p - \mathbf{H}_p \mathcal{X} \right\|^2 + \sum_{k \in \mathcal{B}} \left\| \mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}_{b_{k+1}}^{b_k}, \mathcal{X}) \right\|_{\mathbf{P}_{b_{k+1}}^{b_k}}^2 + \sum_{(l,j) \in \mathcal{C}} \left\| \mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}_l^{c_j}, \mathcal{X}) \right\|_{\mathbf{P}_l^{c_j}}^2 \right\}$$

  - Linearize, solve, and iterate until time budget is reached
  - Ceres Solver (http://ceres-solver.org/)
  - Utilize sparse matrix solver
- Qualitative discussion on solution quality
  - Numerical stability issues always exists, much worse than vSLAM
    - Good: walking and aerial robots
    - Bad: ground vehicle moving in 2D
    - Failure: constant velocity or pure rotation
  - Downgraded performance in distanced scenes

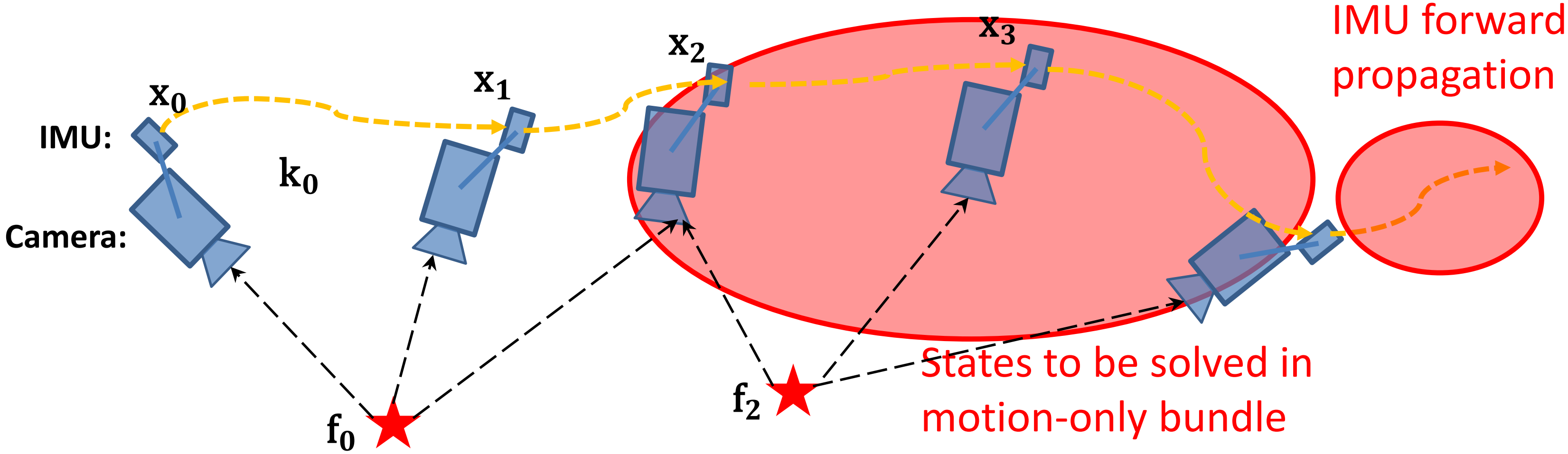
Monocular Visual-Inertial Odometry

- Marginalization
  - Bound computation complexity to a sliding window of states
  - Basic principles:
    - keep as many keyframes with sufficient parallax as possible
    - Maintain matrix sparsity by throwing away visual measurements from non-keyframes



Monocular Visual-Inertial Odometry

- Speeding up
  - The full monocular visual-inertial bundle adjustment runs at 10Hz
  - Use motion-only visual-inertial bundle to boost to 30Hz
  - Use IMU forward propagation to boost to 100Hz

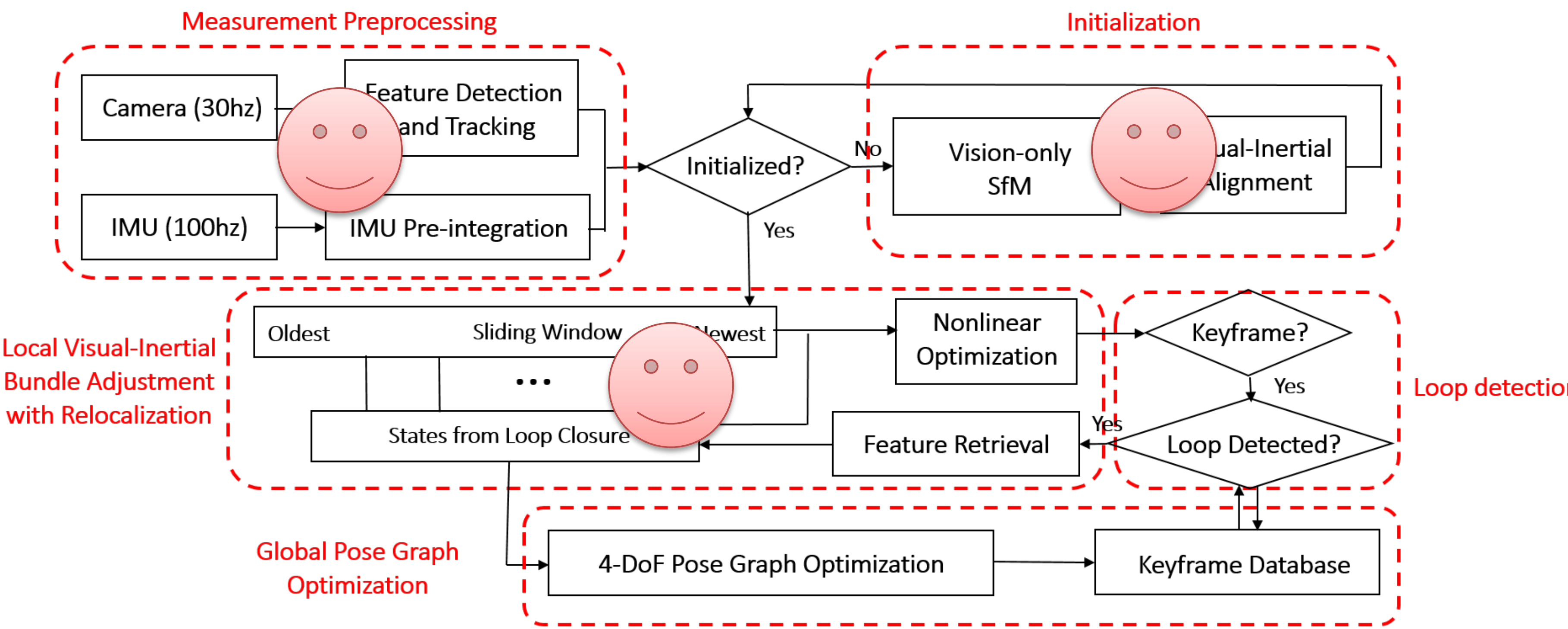


Monocular Visual-Inertial Odometry

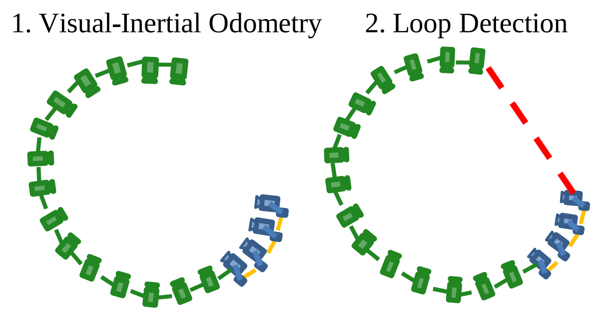
- Failure detection
  - Few trackable feature in the current frame
  - Large jumps in nonlinear solver
  - Abnormal bias or extrinsic parameter calibration
  - Modeled as a standalone module, more to be added…
- Failure recovery
  - Just run the initialization again…
  - Lots of book keeping…

Quick Review

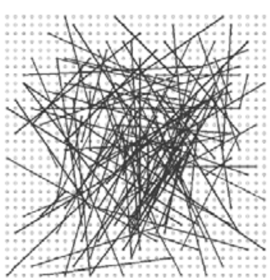
- System diagram

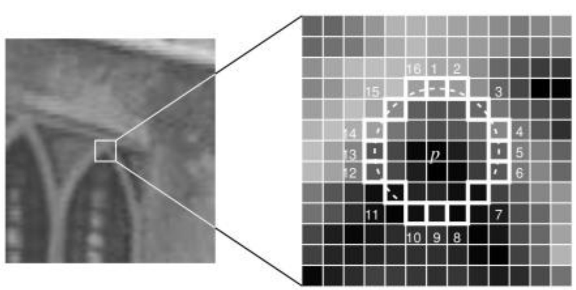


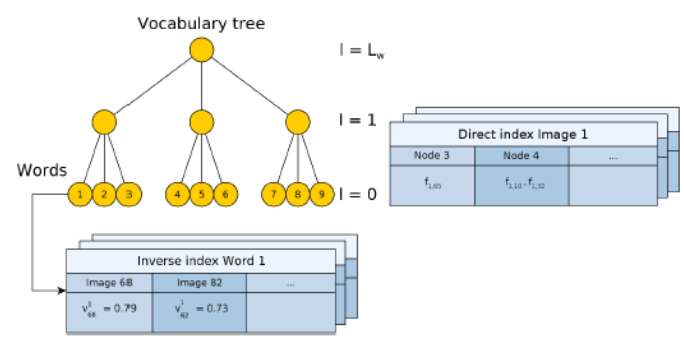
Loop Closure



- Loop detection
  - Describe features by BRIEF
    - Features that we use in the VIO
      (200, not enough for loop detection)
    - Extract new FAST features
      (500, only use for loop detection)
  - Query Bag-of-Word (DBoW2)
    - Return loop candidates



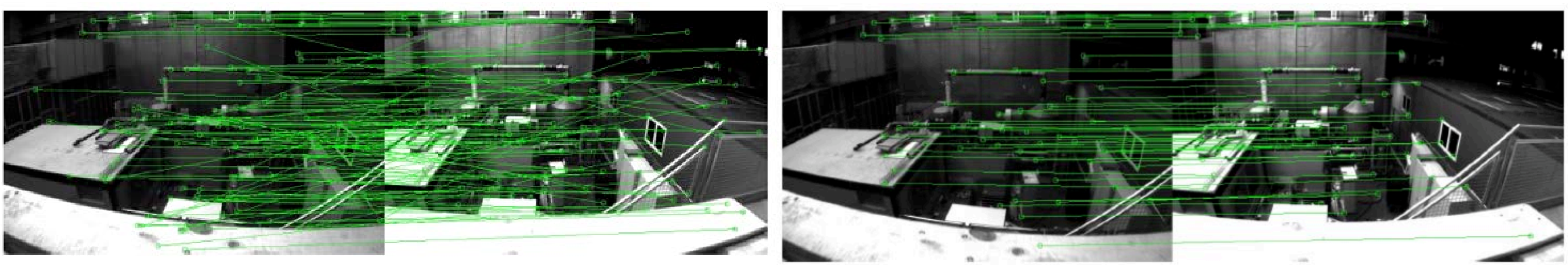




Calonder, Michael, et al. "Brief: Binary robust independent elementary features." *Computer Vision–ECCV 2010* (2010): 778-792.
Gálvez-López, Dorian, and Juan D. Tardos. "Bags of binary words for fast place recognition in image sequences." *IEEE Transactions on Robotics* 28.5 (2012): 1188-1197.

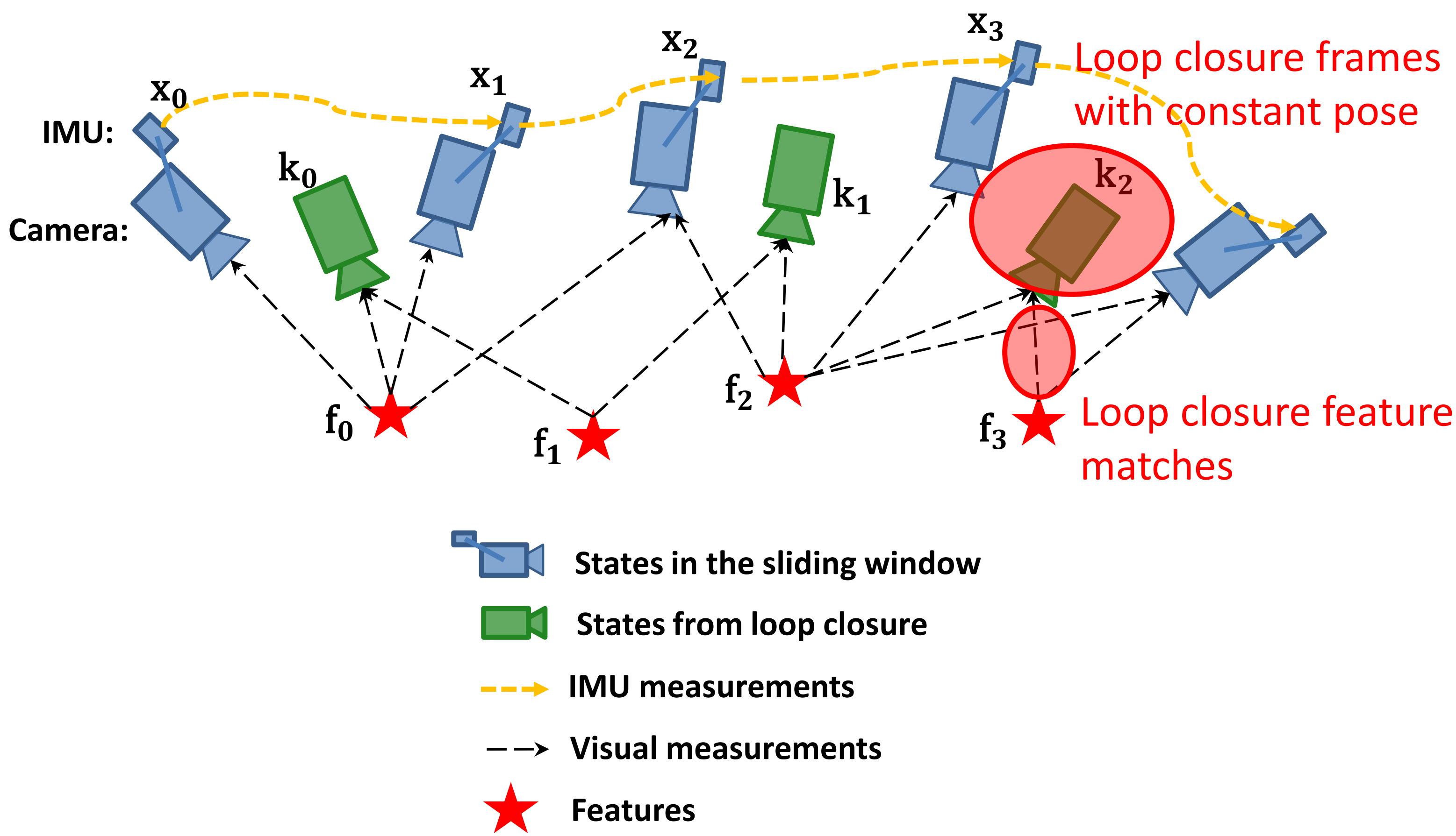
Loop Closure

- Feature Retrieving
  - Try to retrieve matches for features (200) that are used in the VIO
  - BRIEF descriptor match
  - Geometric check
    - Fundamental matrix test with RANSAC
    - At least 30 inliers
- Output:
  - Loop closure frames with known pose
  - Feature matches between VIO frames and loop closure frames



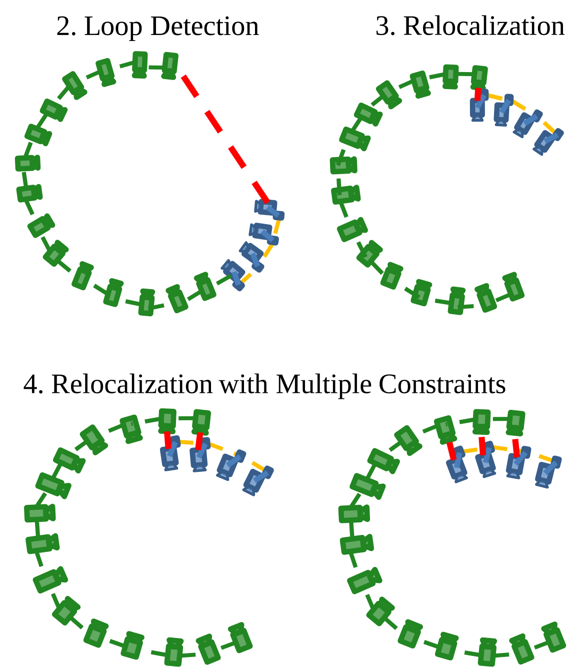
(a) (b)

Monocular Visual-Inertial Odometry with Relocalization



Monocular Visual-Inertial Odometry with Relocalization

- Relocalization
  - Visual measurements for tightly-coupled relocalization
    - Observation of retrieved features in loop closure frames
    - Poses of loop closure frames are constant
    - No increase in state vector dimension for relocalization
    - Allows multi-constraint relocalization



$$\min_{\mathcal{X}} \left\{ \left\| \mathbf{r}_p - \mathbf{H}_p \mathcal{X} \right\|^2 + \sum_{k \in \mathcal{B}} \left\| \mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}^{b_k}_{b_{k+1}}, \mathcal{X}) \right\|^2_{\mathbf{P}^{b_k}_{b_{k+1}}} + \sum_{(l,j) \in \mathcal{C}} \left\| \mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}^{c_j}_l, \mathcal{X}) \right\|^2_{\mathbf{P}^{c_j}_l} + \sum_{(l,v) \in \mathcal{L}} \left\| \mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}^{v}_l, \mathcal{X}, \hat{\mathbf{q}}^w_v, \hat{\mathbf{p}}^w_v) \right\|^2_{\mathbf{P}^{c_v}_l} \right\}, \quad (22)$$

VIO residuals

Loop closure vision measurement residual

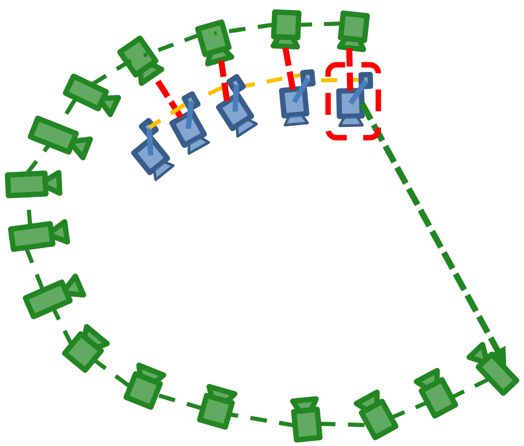
Global Pose Graph SLAM

- 4-DOF pose graph
  - Roll and pitch are observable from VIO
- Adding keyframes into pose graph

$$\hat{\mathbf{p}}_{ij}^{i} = \hat{\mathbf{R}}_{i}^{w^{-1}}(\hat{\mathbf{p}}_{j}^{w} - \hat{\mathbf{p}}_{i}^{w})$$
$$\hat{\psi}_{ij} = \hat{\psi}_{j} - \hat{\psi}_{i}$$

  - Sequential edges from VIO
    - Connected with 4 previous keyframes
  - Loop closure edges
    - Only added when a keyframe is marginalized out from the sliding window VIO
    - Multi-constraint relocalization helps eliminating false loop closures

5. Add Keyframe into Pose Graph



Global Pose Graph SLAM

- 4-DOF relative pose residual:

Observable attitude from VIO

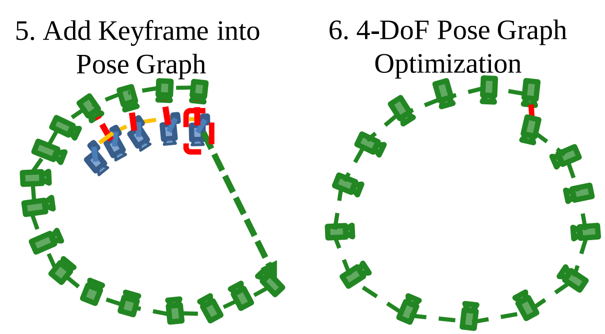
$$\mathbf{r}_{i,j}(\mathbf{p}_i^w, \psi_i, \mathbf{p}_j^w, \psi_j) = \begin{bmatrix} \mathbf{R}(\hat{\phi}_i, \hat{\theta}_i, \psi_i)^{-1}(\mathbf{p}_j^w - \mathbf{p}_i^w) - \hat{\mathbf{p}}_{ij}^i \\ \psi_j - \psi_i - \hat{\psi}_{ij} \end{bmatrix}$$

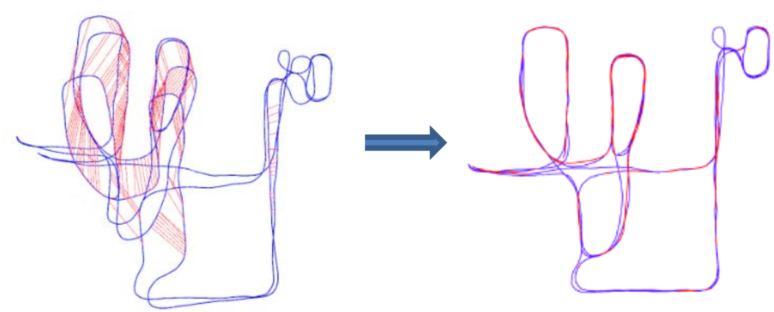
- Minimize the following cost function
  - Sequential edge from VIO
  - Loop closure edges
    - Huber norm for rejection of wrong loops

Sequential edges

$$\min_{\mathbf{p},\psi} \left\{ \sum_{(i,j)\in\mathcal{S}} \|\mathbf{r}_{i,j}\|^2 + \sum_{(i,j)\in\mathcal{L}} h(\|\mathbf{r}_{i,j}\|) \right\}$$

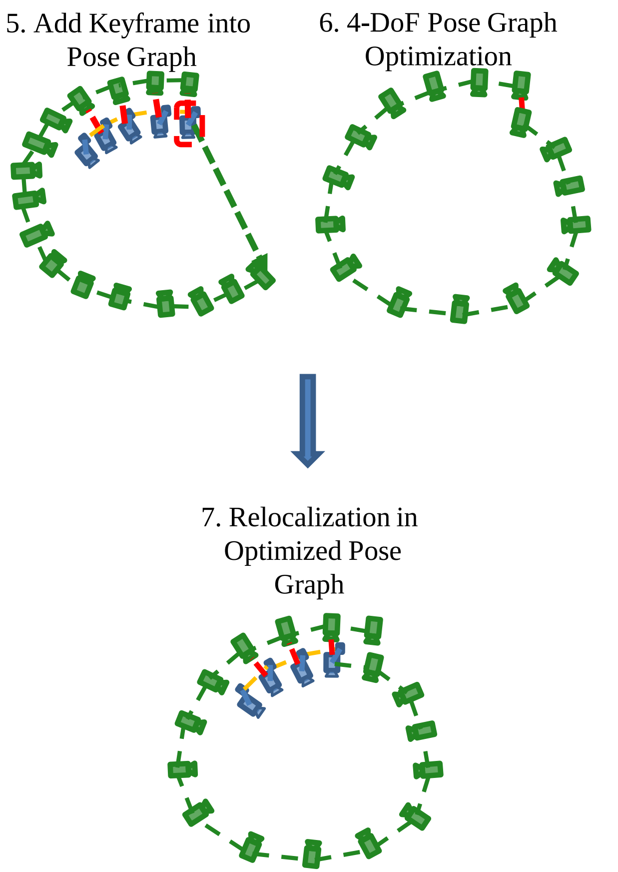
Loop closure edges





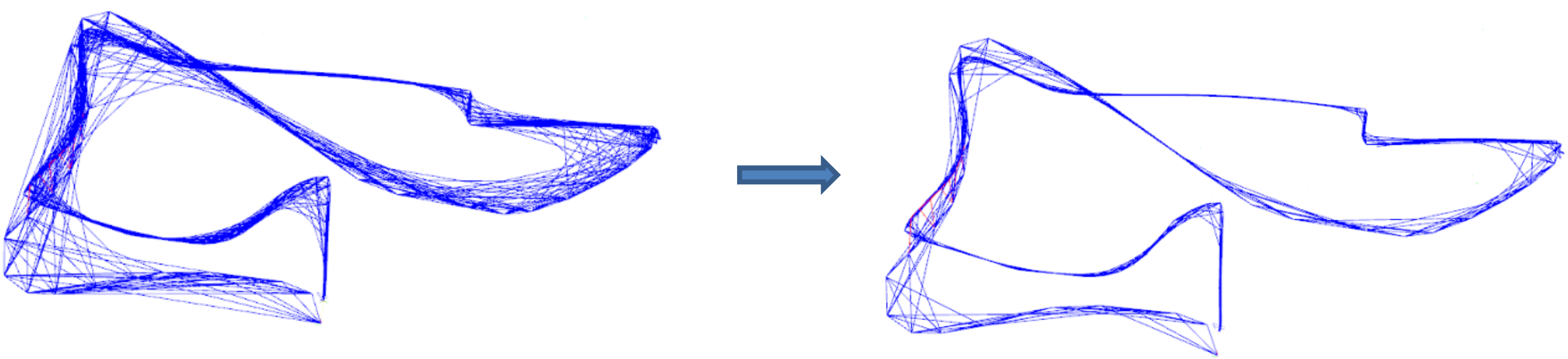
Global Pose Graph SLAM

- More on relocalization
  - Relocalization continued on the optimized pose graph
  - Relocalization and pose graph optimization run in different threads and in different rate
  - Pose graph optimization can be very slow for large-scale environments



Global Pose Graph SLAM

- Simple strategy for pose graph sparsification
  - All keyframes with loop closure constraints will be kept
  - Other keyframes that are either too close to its neighbors or have very similar orientations will be removed



Visual-Inertial SLAM for Autonomous Drone

Monoculor Visual-Inertial System (VINS-Mono) on MAV Platform for Automous Flight

Tong Qin, Peiliang Li, Zhenfei Yang and Shaojie Shen

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

HKUST
Aerial Robotics Group

Open source: https://github.com/HKUST-Aerial-Robotics/VINS-Mono

Visual-Inertial SLAM in Large-Scale Environment

Monoculor Visual-Inertial System (VINS-Mono)
Indoor and Outdoor Performance

Tong Qin, Peiliang Li, Zhenfei Yang and Shaojie Shen

香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

HKUST
Aerial Robotics Group

Open source: https://github.com/HKUST-Aerial-Robotics/VINS-Mono

Visual-Inertial SLAM for Mobile AR

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

VINS-Mobile: Monocular Visual-Inertial State Estimator on Mobile Phones

HKUST Aerial Robotics Group
uav.ust.hk

No plots, only code…

https://github.com/HKUST-Aerial-Robotics

Open Source

HKUST Aerial Robotics Group

CYT 2014A, HKUST, Clear Water Bay, Kowloo... http://uav.ust.hk/

Repositories | People 8

Search repositories... | Type: All | Language: All

VINS-Mono

A Robust and Versatile Monocular Visual-Inertial State Estimator

C++ ★ 222 137 Updated 4 hours ago

VINS-Mobile

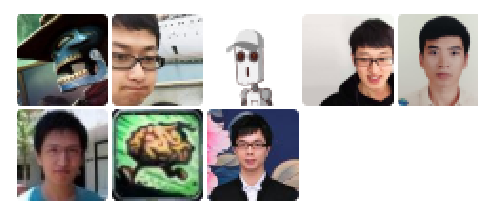
Monocular Visual-Inertial State Estimator on Mobile Phones

C++ ★ 350 201 Updated 8 days ago

Top languages

C++

People 8



HoloKit
(developed by Amber Garage using VINS-Mobile)

HoloKit

View More by This Developer

By Amber Garage

Open iTunes to buy and download apps.

View in iTunes

Free

Category: Entertainment
Updated: Jun 06, 2017
Version: 1.2
Size: 314 MB
Language: English
Seller: Amber Garage, Inc.
© 2017 Amber Garage Inc.
Rated 4+

Compatibility: Requires iOS 10.2 or later. Compatible with iPhone, iPad, and iPod touch.

Customer Ratings

We have not received enough ratings to display an average for the current version of this application.

Description

HoloKit takes you into a funny Mixed Reality experience, where virtual objects merge into real world. Powered by the accurate gyro and camera on your iPhone, HoloKit solidly places virtual objects onto your table or floor, as if they were physically there. In addition, have fun with the magic that these objects will stay there even if you walk to

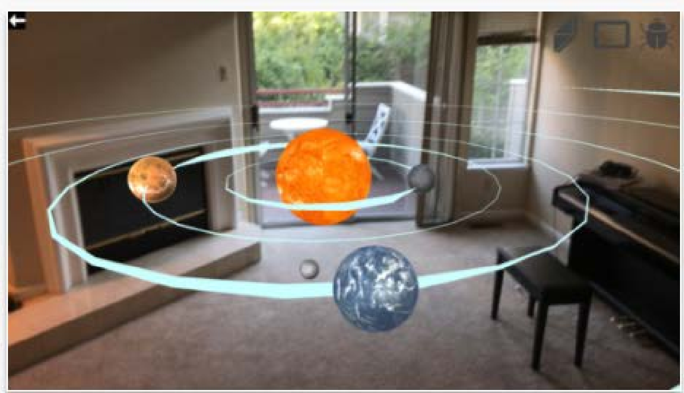
Amber Garage Web Site ▸ HoloKit Support ▸ ...More

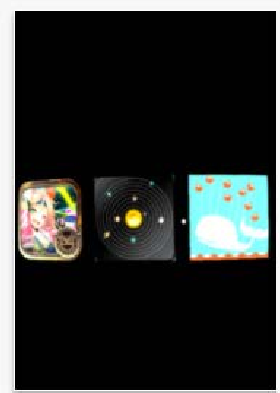
What's New in Version 1.2

Added 4 experiences:
Solar Sytem
Whale Jump

...More

iPhone Screenshots



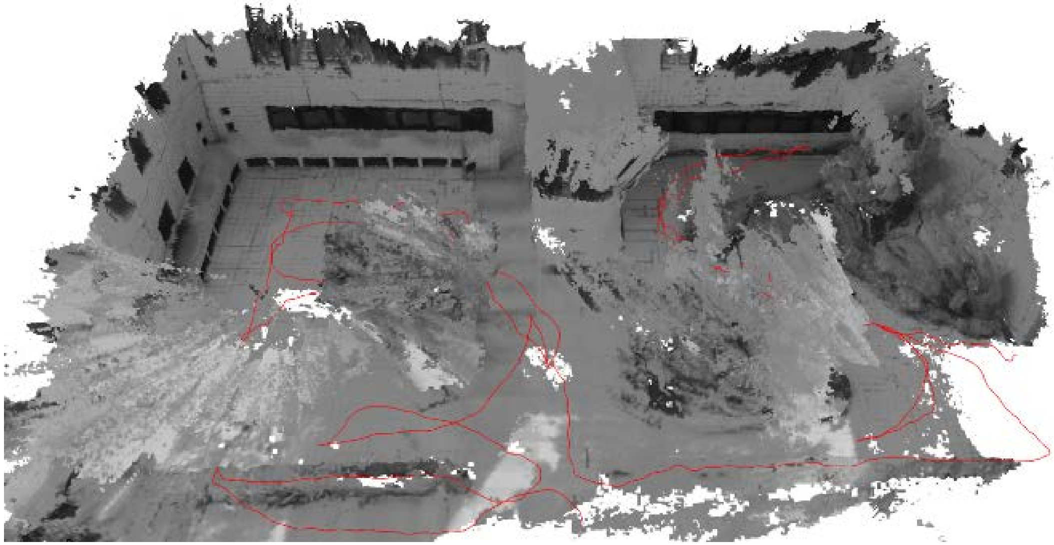


Remarks on Monocular Visual-Inertial State Estimation

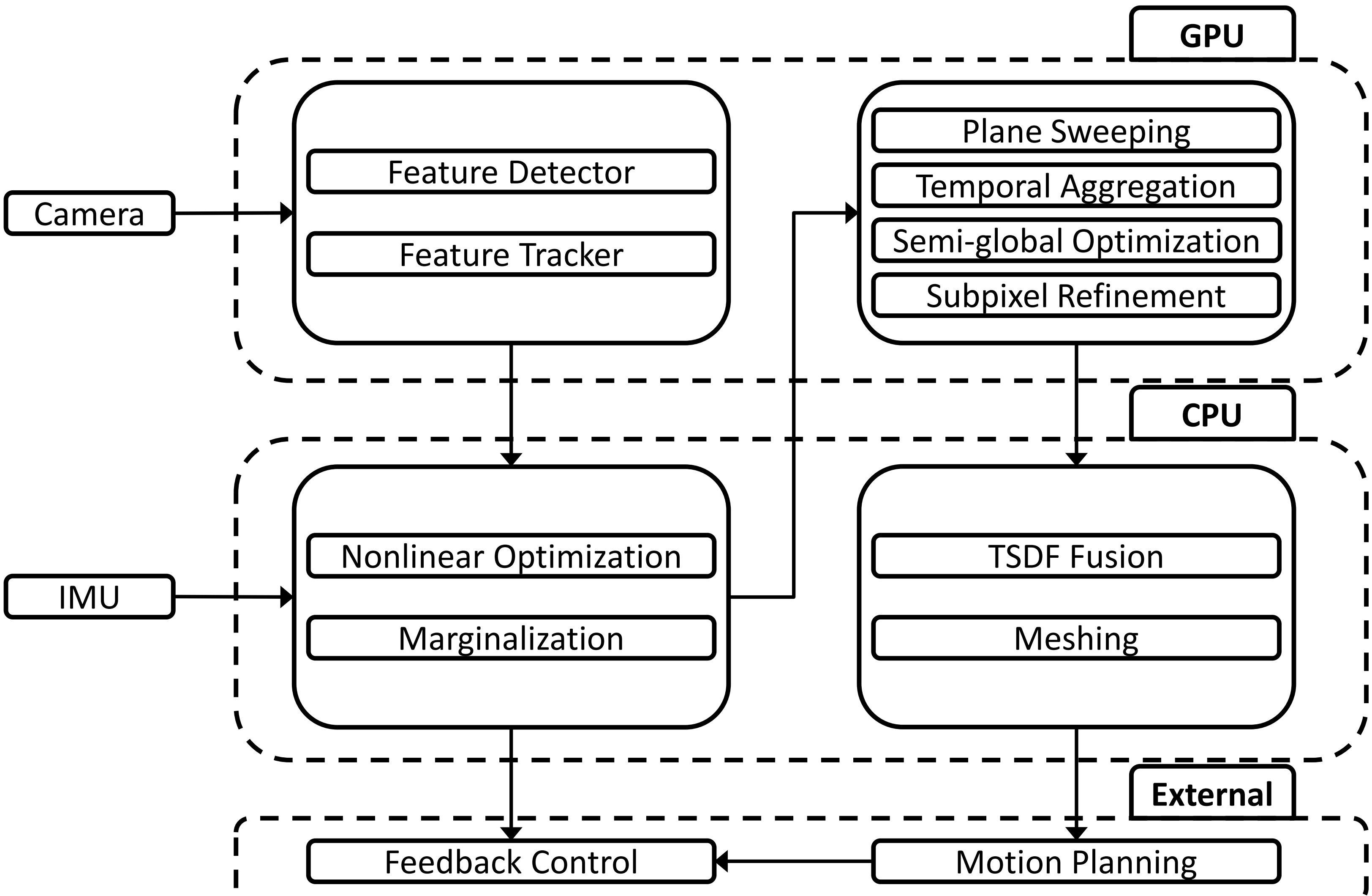
- Important factors
  - Tightly-coupled formulation
  - Global shutter camera
  - Sensor synchronization and timestamps
  - Camera-IMU rotation
  - Estimator initialization
- Not-so-important factors
  - Camera-IMU translation
  - Types of features (we use the simplest corner+KLT)
  - Quality of feature tracking (outlier is acceptable)
- Failures
  - Long range scenes (aerial vehicles)
  - Constant velocity (ground vehicle)
  - Pure rotation (augmented reality)

A Quick Overview on Real-Time Monocular Visual-Inertial Dense Mapping

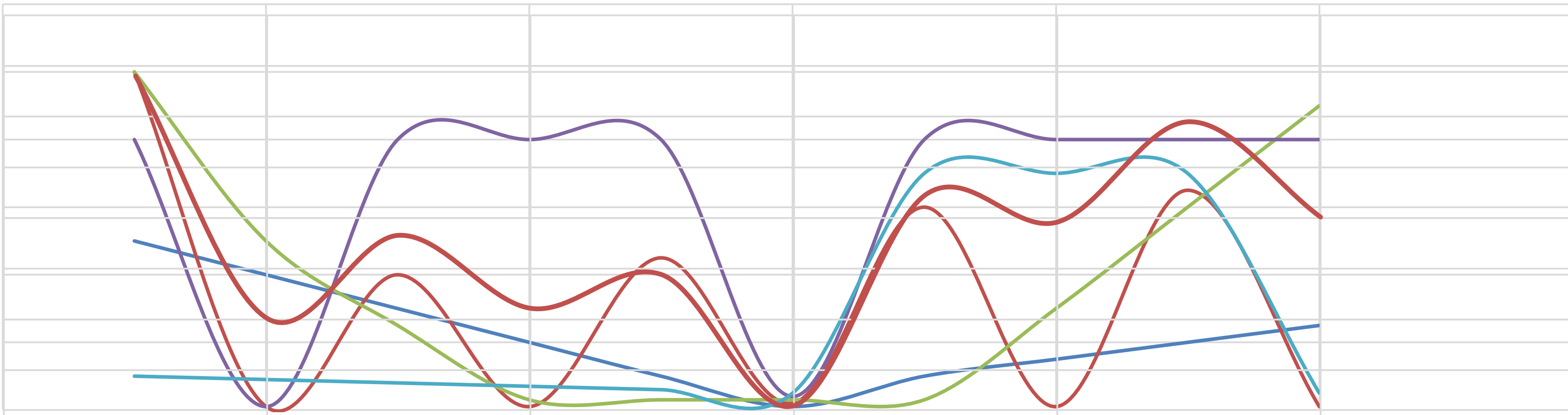
- Design considerations
  - No dense joint optimization
  - Condition on accurate feature-based visual-inertial SLAM
  - Use as many frames as possible
  - Parallelization on GPU

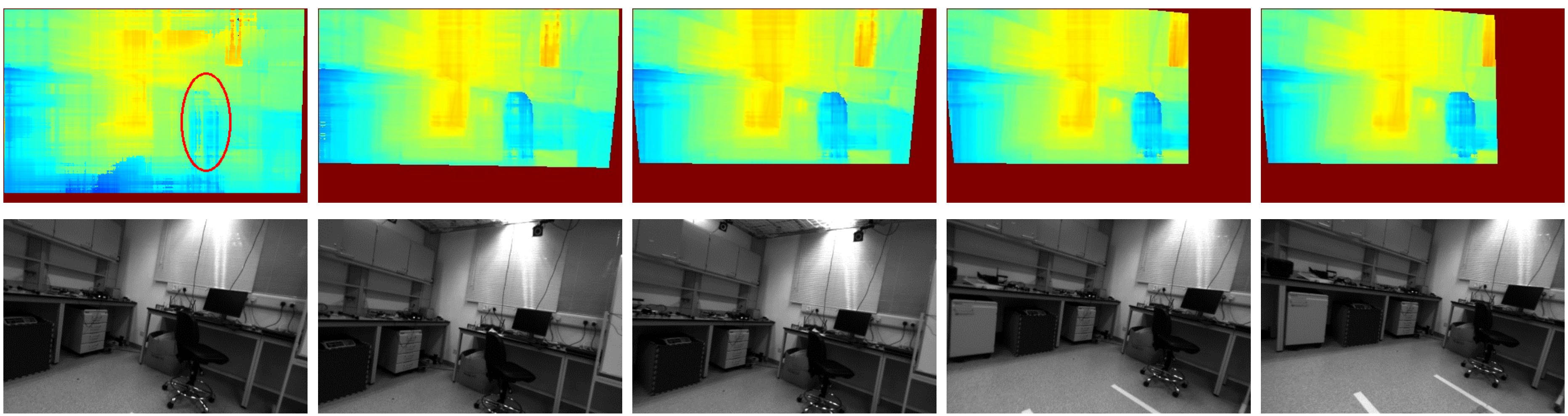


Monocular Visual-Inertial Dense Mapping

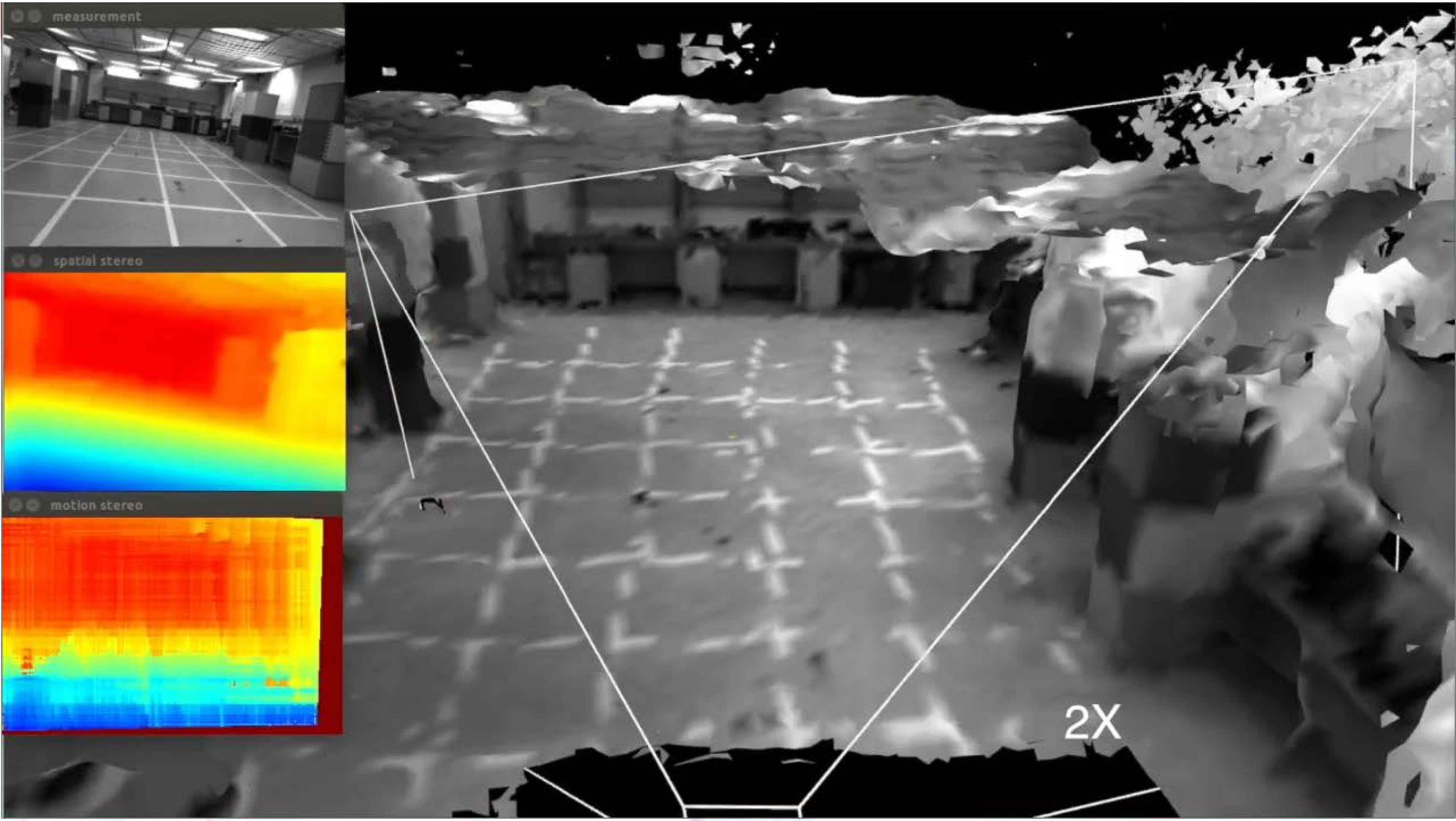


Multi-View Plane Sweeping Motion Stereo





Volumetric Fusion



Handheld Dense Mapping

Autonomous Aerial Navigation

Autonomous Aerial Navigation

Final Remarks

- IMU is great!!!
- Feature-based visual-inertial SLAM is very close to done
  - Some research work remains:
    - Online observability analysis
    - Large-scale, long duration operations
    - Extreme environments
    - Extreme motions
  - Big engineering challenges towards mass deployment on different devices (Android phones?)
    - Intrinsic and extrinsic calibration of IMU, rolling shutter, etc.
    - Synchronization issues
    - Poor sensors and manufacturing variations
    - Insufficient computating power
  - Big players are moving in

Final Remarks

- Real-time dense mapping is interesting
  - Very few working implementations
  - How to reduce computation?
  - Joint optimization or alternating estimation?
  - Textureless and repetitive patterns
  - Combination of learning and geometric-based methods
  - Efficient map representation for large-scale environments

Thanks!

Questions?