

Introduction to Aerial Robotics

Lecture 10

Shaojie Shen

Associate Professor

Dept. of ECE, HKUST



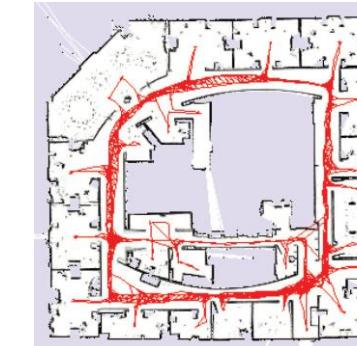
25 April 2023

Outline

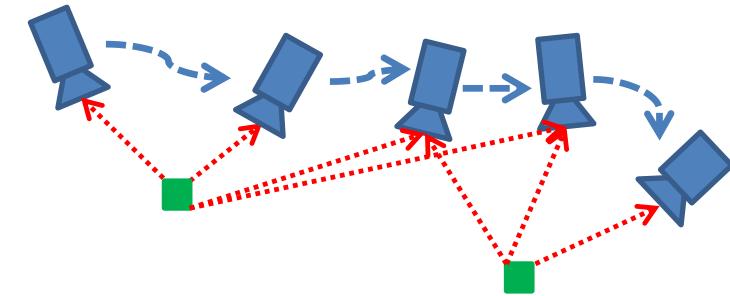
- The Basics

$$P(x | y, z) = \frac{P(y | x, z) P(x | z)}{P(y | z)}$$

- 2D Pose Graph SLAM



- Monocular Visual-Inertial
SLAM



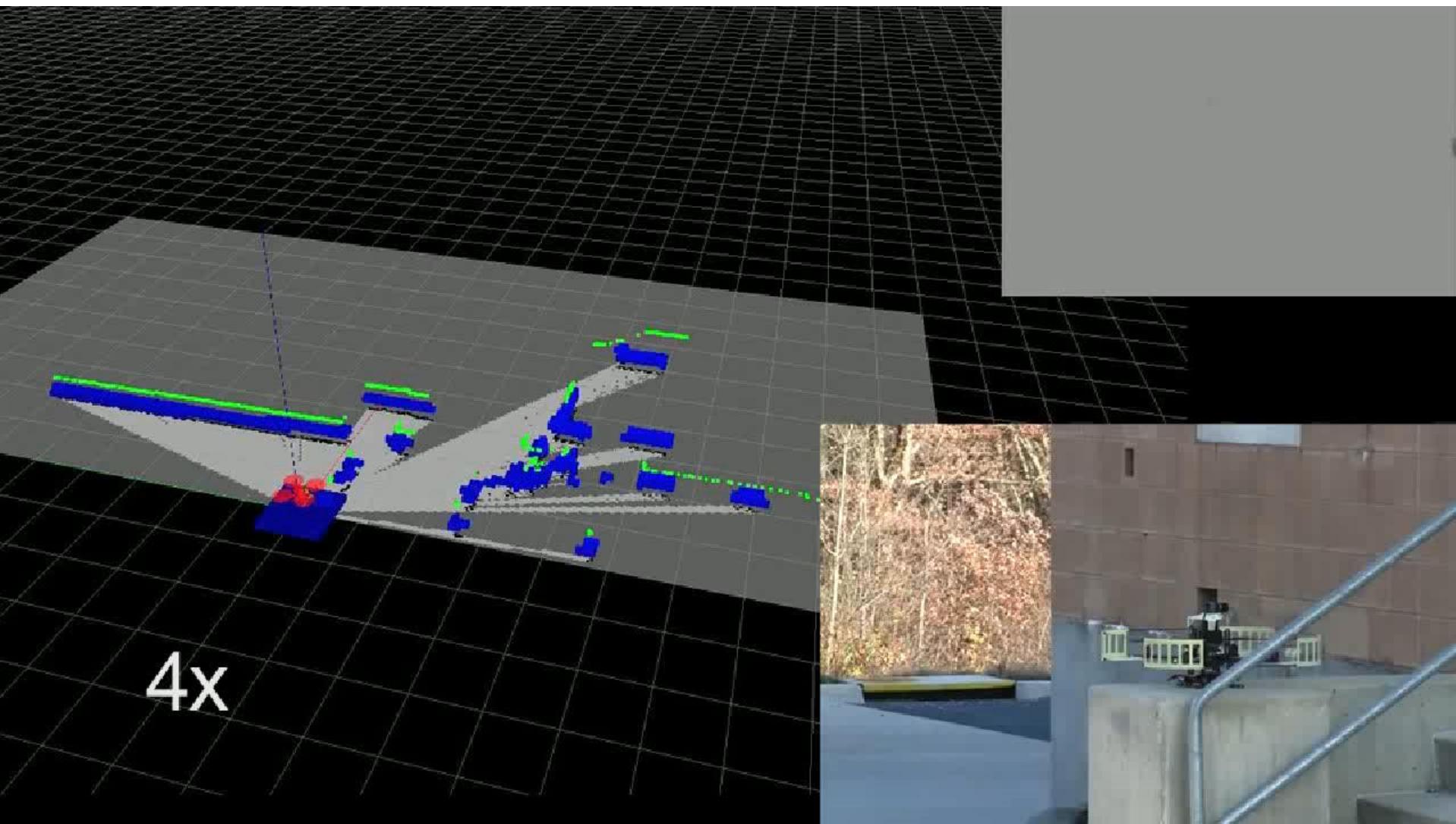
This is Science Fiction...



From Movie “Prometheus”



This is Real...



The Localization Problem

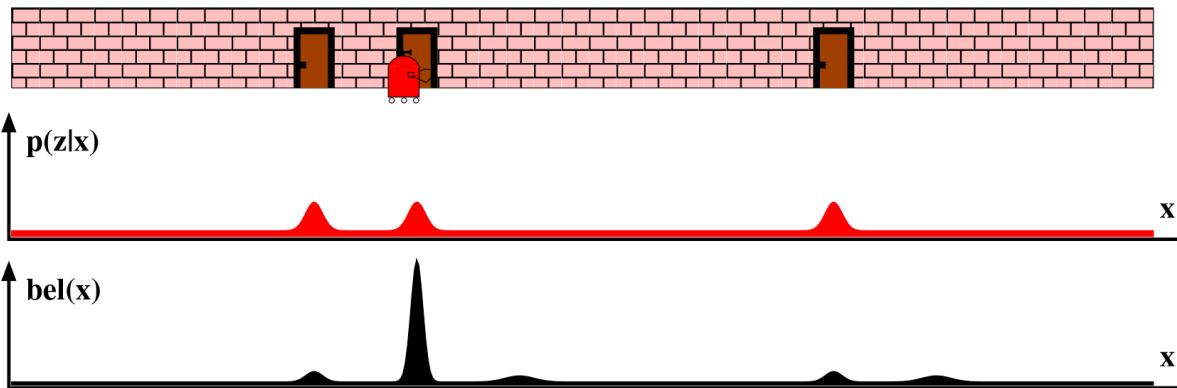
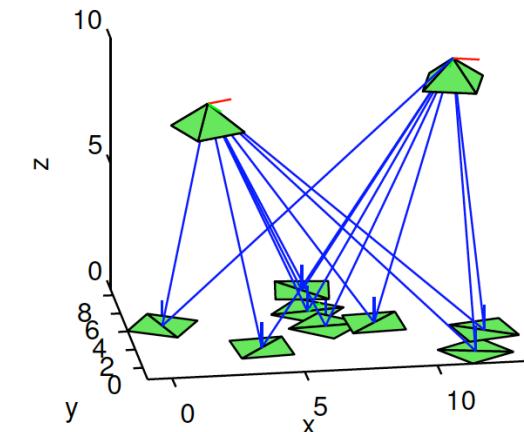
A robot is exploring a known, static environment.

Given:

- Measurements from sensor(s)

Estimate:

- Path/trajectory of the robot



The SLAM Problem

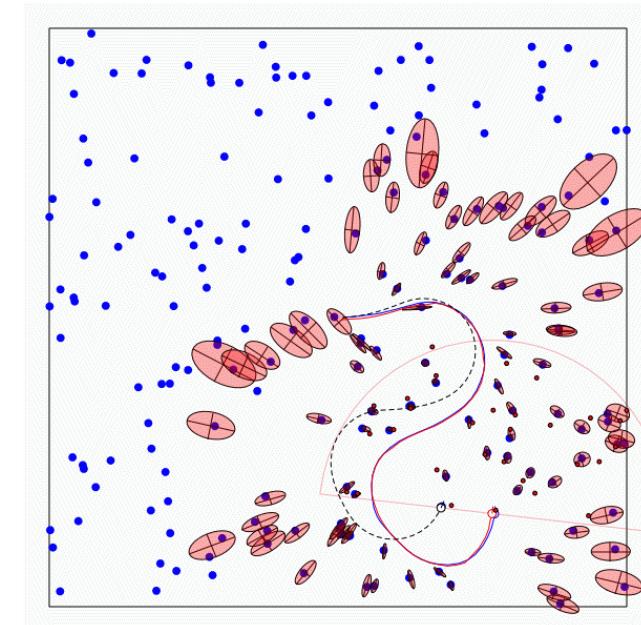
A robot is exploring an unknown, static environment.

Given:

- Measurements from sensor(s)

Estimate:

- The map
- Path/trajectory of the robot



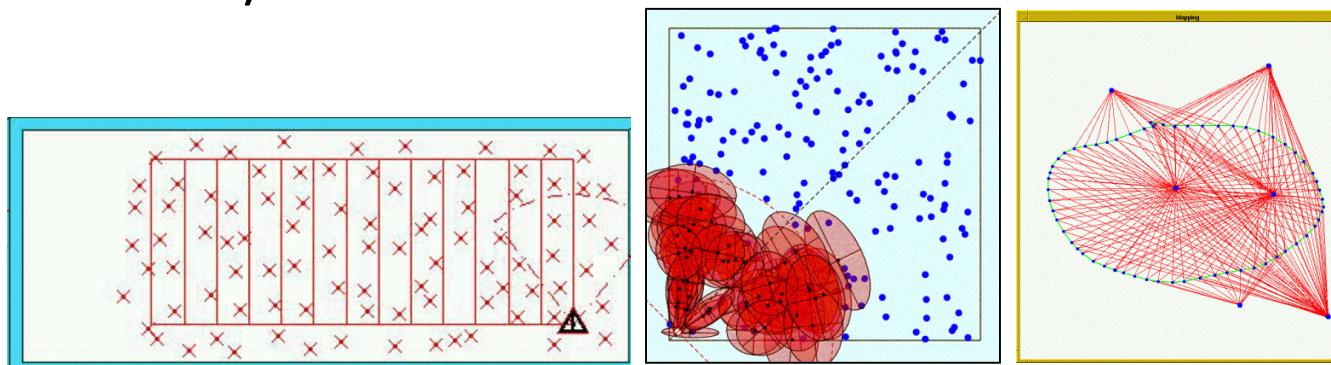
Representations

- Grid maps or scans



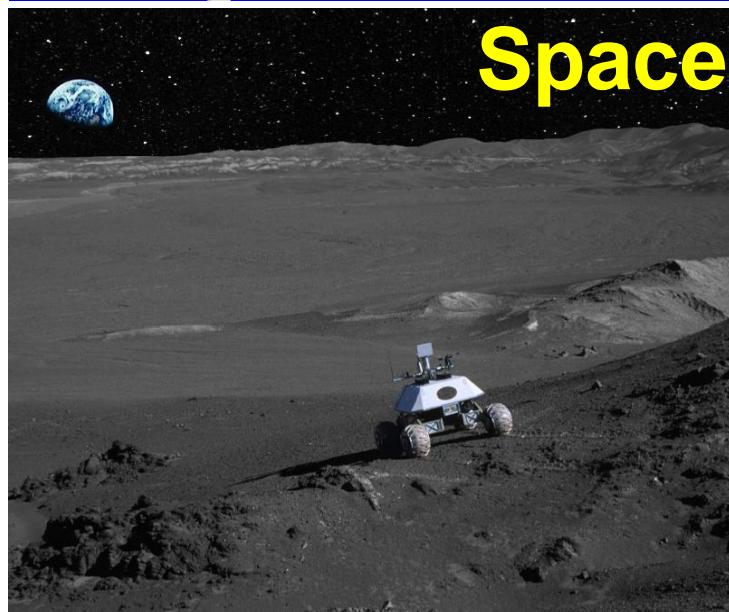
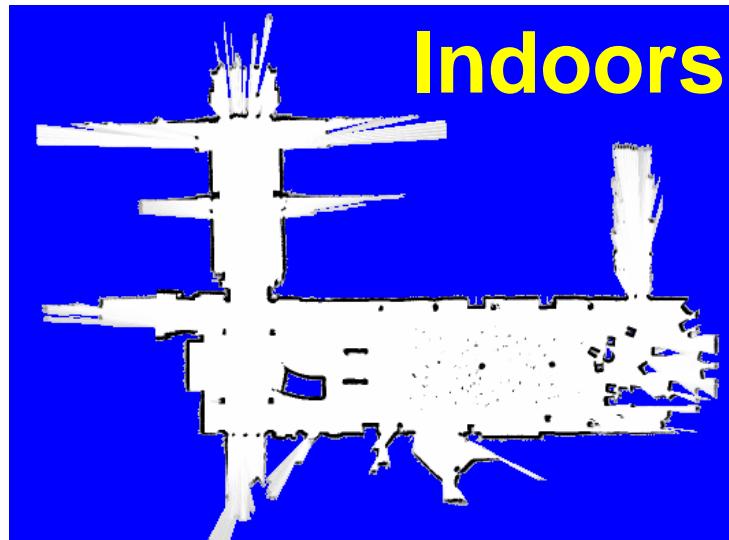
[Lu & Milios, 97; Gutmann, 98; Thrun 98; Burgard, 99; Konolige & Gutmann, 00; Thrun, 00; Arras, 99; Haehnel, 01;...]

- Landmark/feature-based



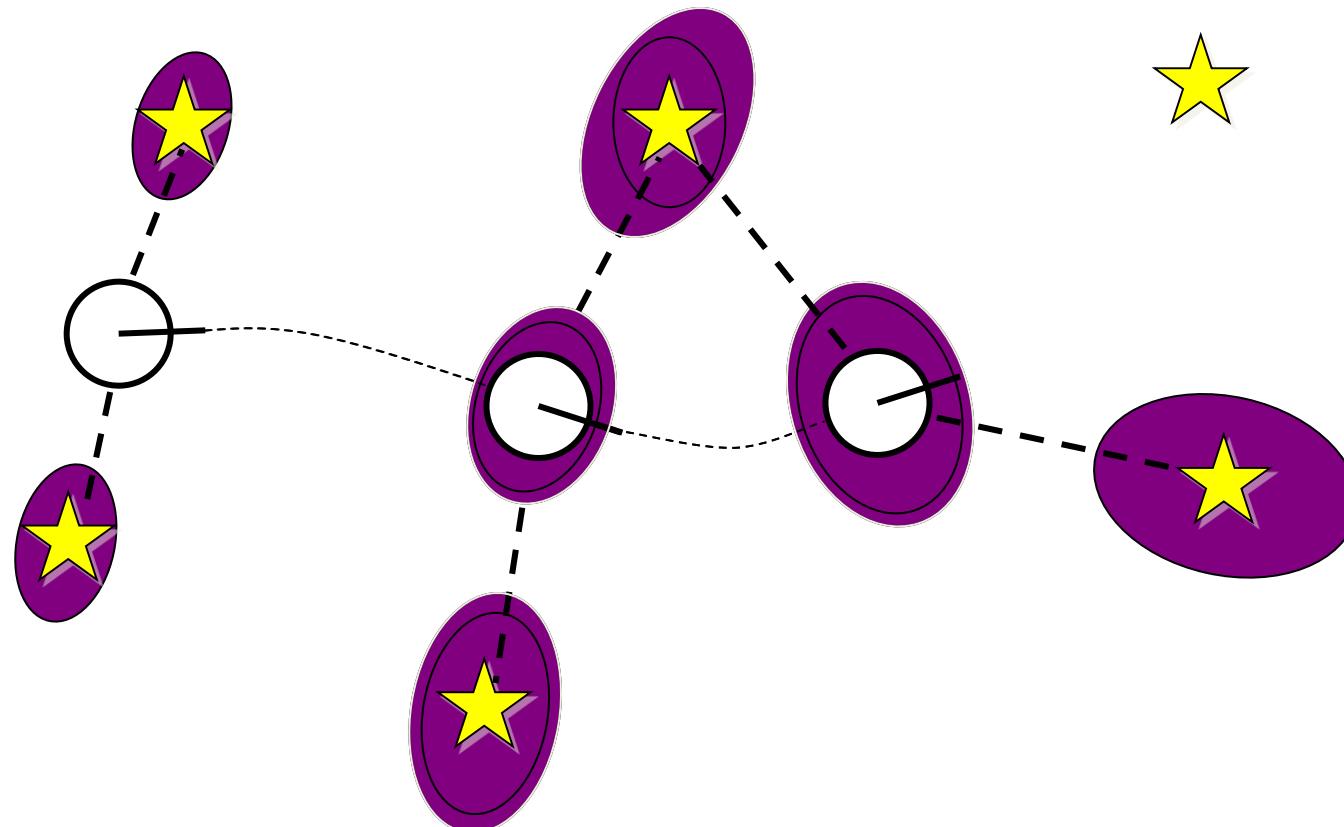
[Leonard et al., 98; Castelanos et al., 99; Dissanayake et al., 2001; Montemerlo et al., 2002;...]

SLAM Applications



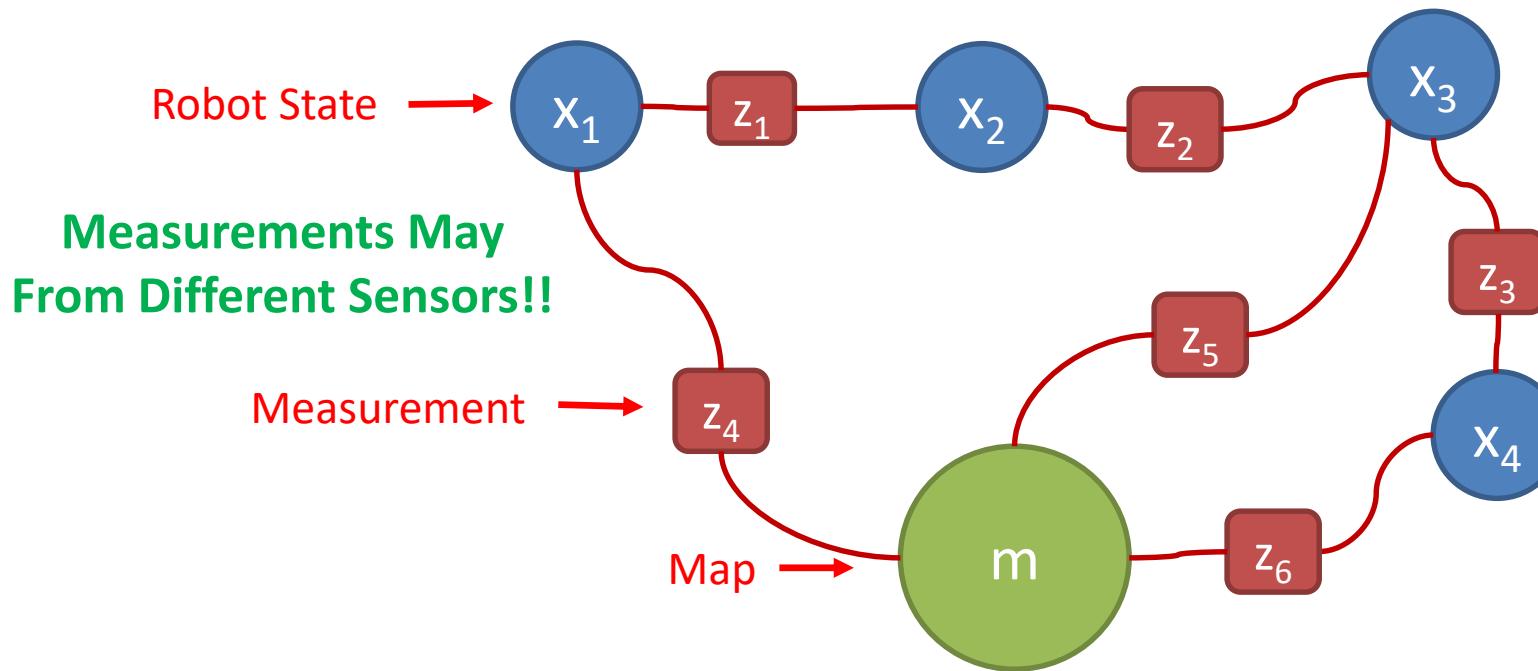
Why is SLAM a hard problem?

SLAM: robot path and map are both **unknown**



Robot path error correlates errors in the map

Graphical Model of SLAM

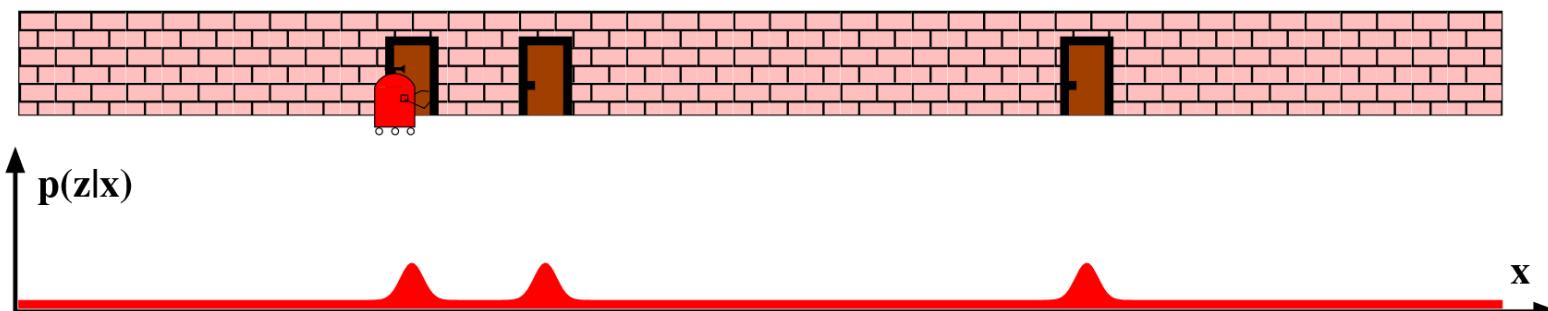


$$(\hat{x}_{1:t}, \hat{m}) = \arg \max_{x_{1:t}, m} p(x_{1:t}, m | z_{1:N})$$

??????

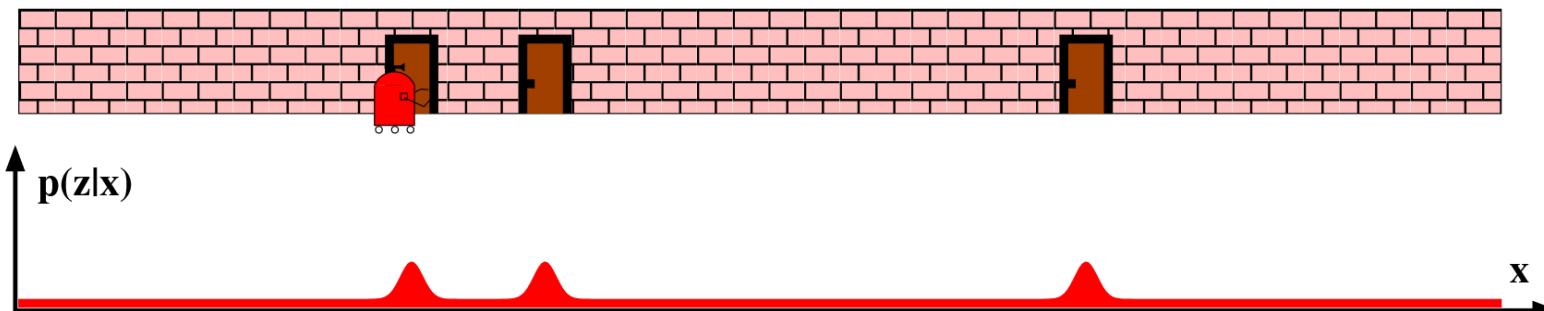
Conditional Probability

- **Definition:** Probability of an event z occurring conditioned on another event x occurring
- $p(z | x) = \frac{p(x,z)}{p(x)} \Leftrightarrow p(x,z) = p(z | x) p(x)$
- **Example:**
 - $z = \{\text{there is a door next to the robot}\}$



Conditional Independence

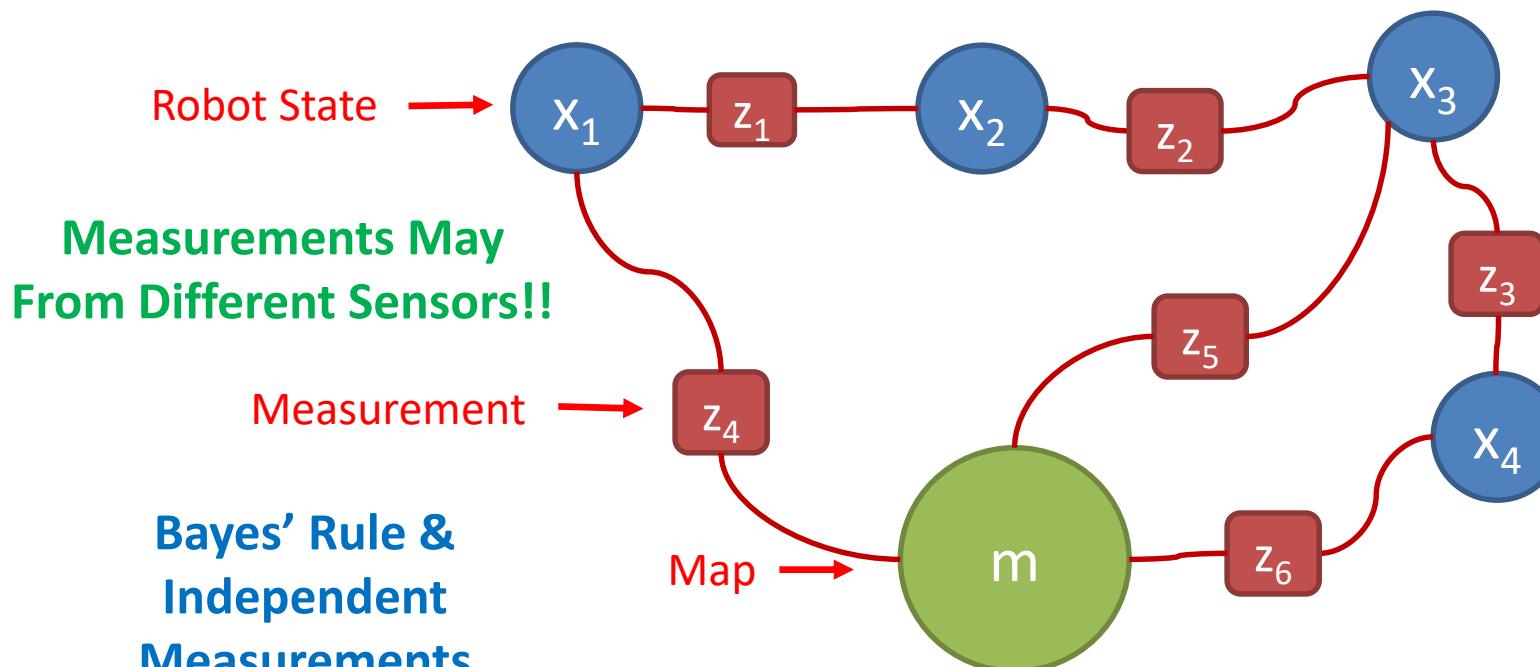
- **Definition:** Two random variables are *conditionally independent* if the outcome of one has *no effect* on the outcome of the other when conditioned on the outcome of a third random variable
- $p(z_1, z_2 | x) = p(z_1 | x) p(z_2 | x)$
- **Example:**
 - Let Z_1, Z_2 be two measurements taken from the same place
 - Z_1, Z_2 are conditionally independent given X



Bayes' Theorem

- $p(x | z) = \frac{p(z | x) p(x)}{p(z)} = \frac{p(z | x) p(x)}{\int p(z | x') p(x') dx'}$
- **Intuition:** Describes how the belief about a random variable X should change to account for the collected evidence (measurement) z
- **Derivation:**
 - $p(x, z) = p(z | x) p(x) = p(x | z) p(z)$

SLAM as Maximum A Posteriori Estimation



$$\begin{aligned}(\hat{x}_{1:t}, \hat{m}) &= \arg \max_{x_{1:t}, m} p(x_{1:t}, m | z_{1:N}) \\&= \arg \max_{x_{1:t}, m} \prod_{i=1 \dots N} \boxed{p(z_i | x_{1:t}, m)} \cdot \boxed{p(x_{1:t}, m)}\end{aligned}$$

Measurement Models Prior

The Gaussian Case

- The multivariate Gaussian distribution

$$p(z | x, m) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} e^{-\frac{1}{2}(z-h(x,m))^T \Sigma^{-1} (z-h(x,m))}$$

Measurement Residual

- The Gaussian case for SLAM:

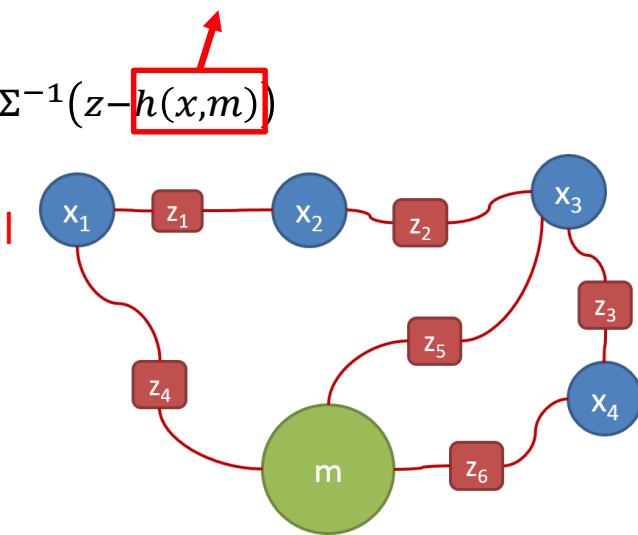
$$\begin{aligned} (\hat{x}_{1:t}, \hat{m}) &= \arg \max_{x_{1:t}, m} p(x_{1:t}, m | z_{1:N}) \\ &= \arg \max_{x_{1:t}, m} \prod_{i=1 \dots N} p(z_i | x_{1:t}, m) \cdot p(x_{1:t}, m) \\ &= \arg \min_{x_{1:t}, m} \sum_{i=1 \dots N} r_{z_i}^T \Sigma_{z_i}^{-1} r_{z_i} + r_{xm}^T \Sigma_{xm}^{-1} r_{xm} \end{aligned}$$

Measurement Residual

$$\boxed{\arg \min_{x_{1:t}, m} \sum_{i=1 \dots N} \|r_{z_i}\|_{\Sigma_{z_i}}^2 + \|r_{xm}\|_{\Sigma_{xm}}^2}$$

Nonlinear
Least Square!

Measurement Model: Predicted measurement from states



SLAM as Nonlinear Least Square

- Nonlinear least square

$$\min_{x_{1:t}} \sum_m \sum_{i=1 \dots N} r_{z_i}^T \Sigma_{z_i}^{-1} r_{z_i} + r_{xm}^T \Sigma_{xm}^{-1} r_{xm}$$

- Linearization

$$r_{z_i} \approx r_{z_i, \hat{X}} + \frac{\partial r_{z_i, \hat{X}}}{\partial X} \Big|_{\hat{X}} \delta X = r_{z_i, \hat{X}} + J_{z_i, \hat{X}} \delta X$$

Measurement
Jacobians

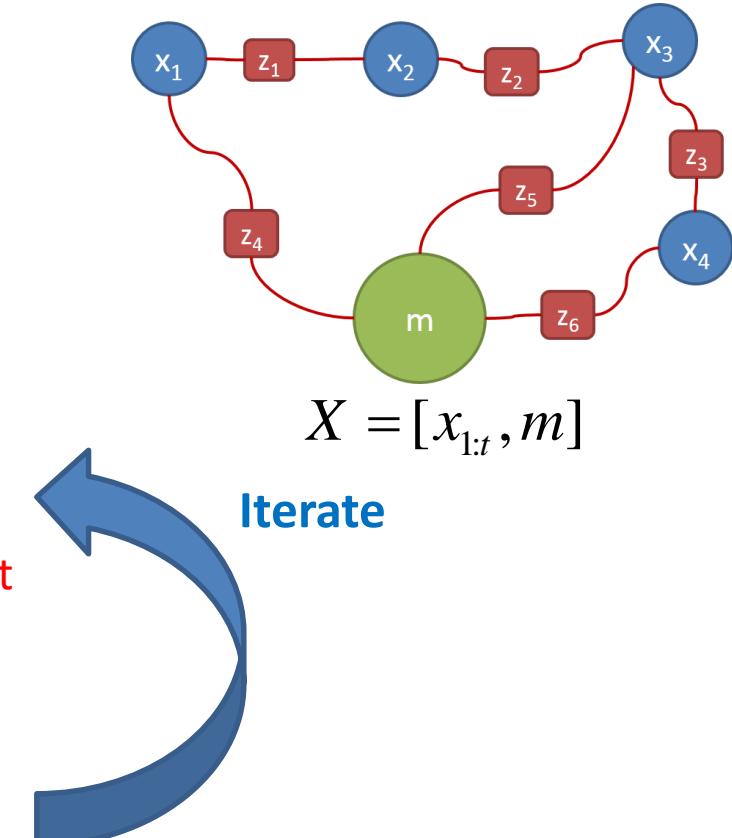
- Gauss-Newton Method

$$J^T \Sigma^{-1} J \delta X = -J^T \Sigma^{-1} r$$

$$\hat{X}^{(k+1)} = \hat{X}^{(k)} + \delta X$$

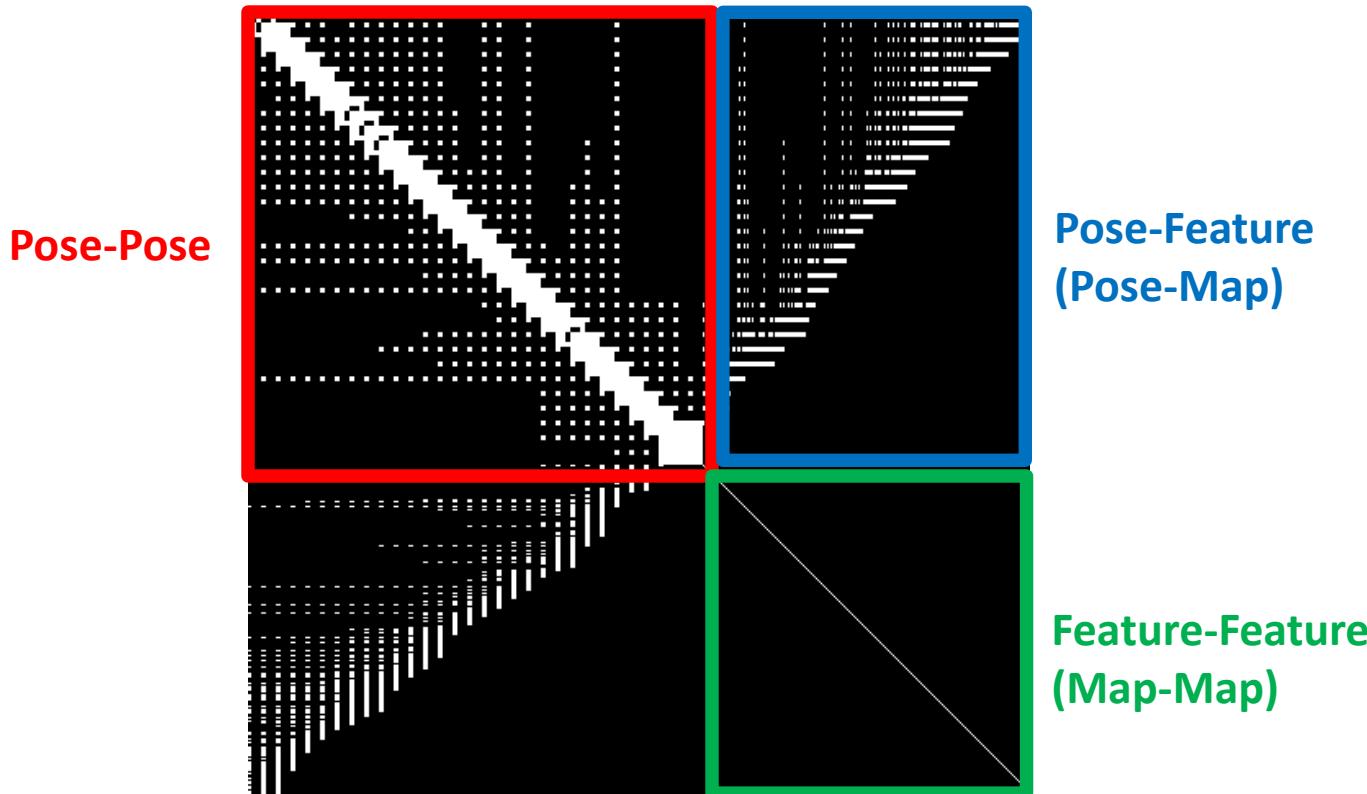
Information
matrix

Stacked Jacobian,
covariance, and residual



SLAM as Nonlinear Least Square

- Linear system - the information matrix
 - Efficient solution via sparse linear equation solver

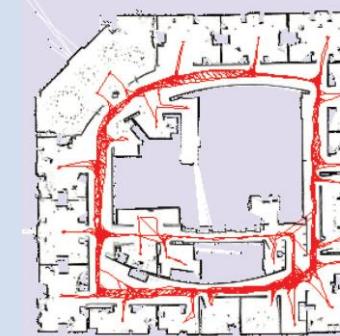


Outline

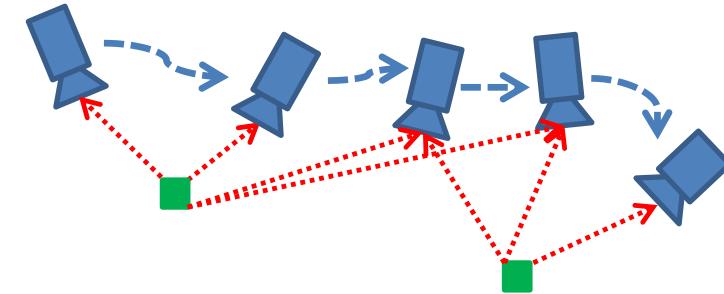
- The Basics

$$P(x | y, z) = \frac{P(y | x, z) P(x | z)}{P(y | z)}$$

- 2D Pose Graph SLAM

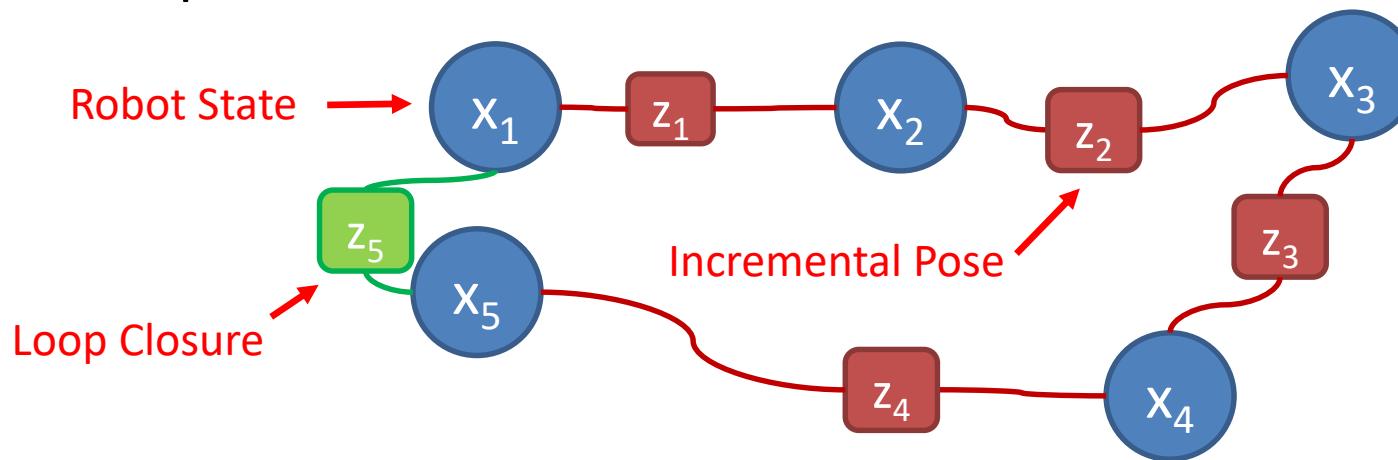


- Monocular Visual-Inertial
SLAM



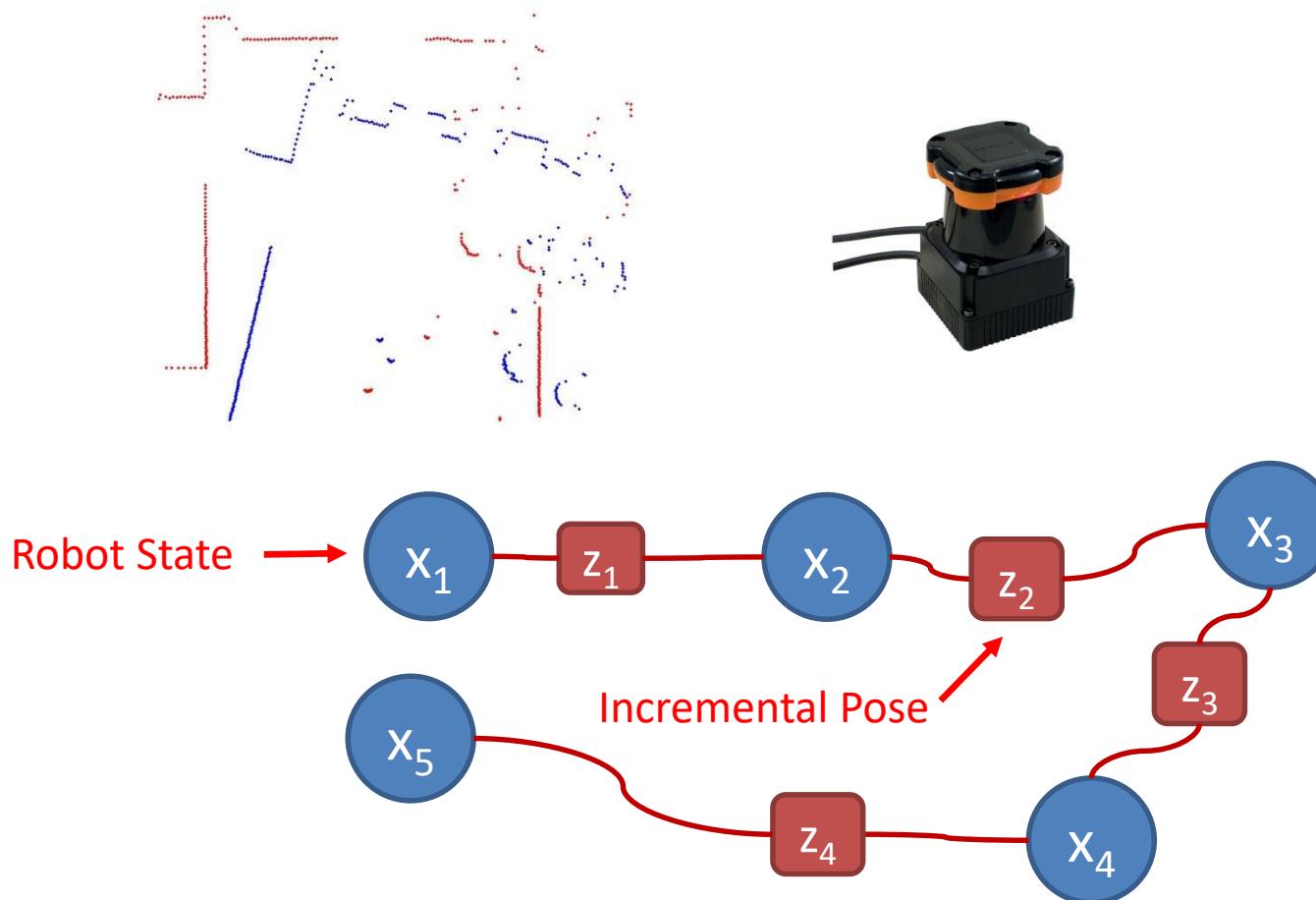
2D Pose Graph SLAM

- The only type of measurement is 2D rigid body transformations between 2D robot poses
- Graphical representation
- No Map



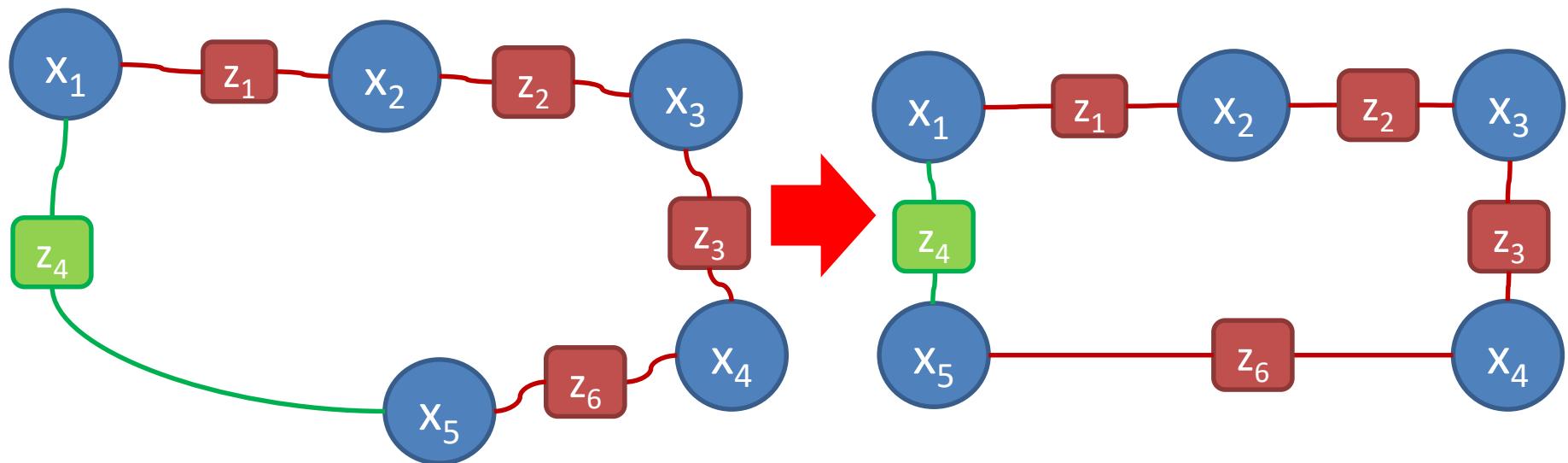
2D Pose Graph SLAM

- Incremental pose transformation from scan matching



2D Pose Graph SLAM

- Global consistency enforced by loop closure

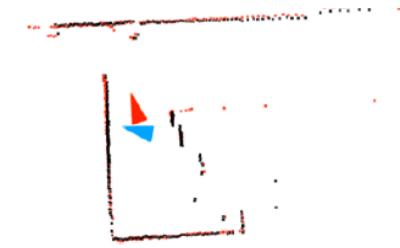
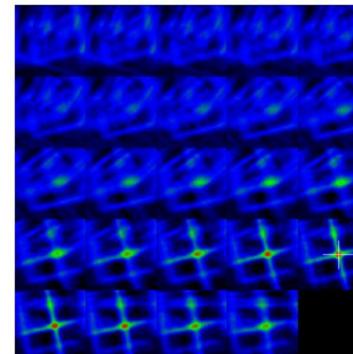


Loop Closure

- Detect re-visiting of places, and find out relative poses
 - Appearance-based: FABMAP (M. Cummins and P. Newman, 2008)

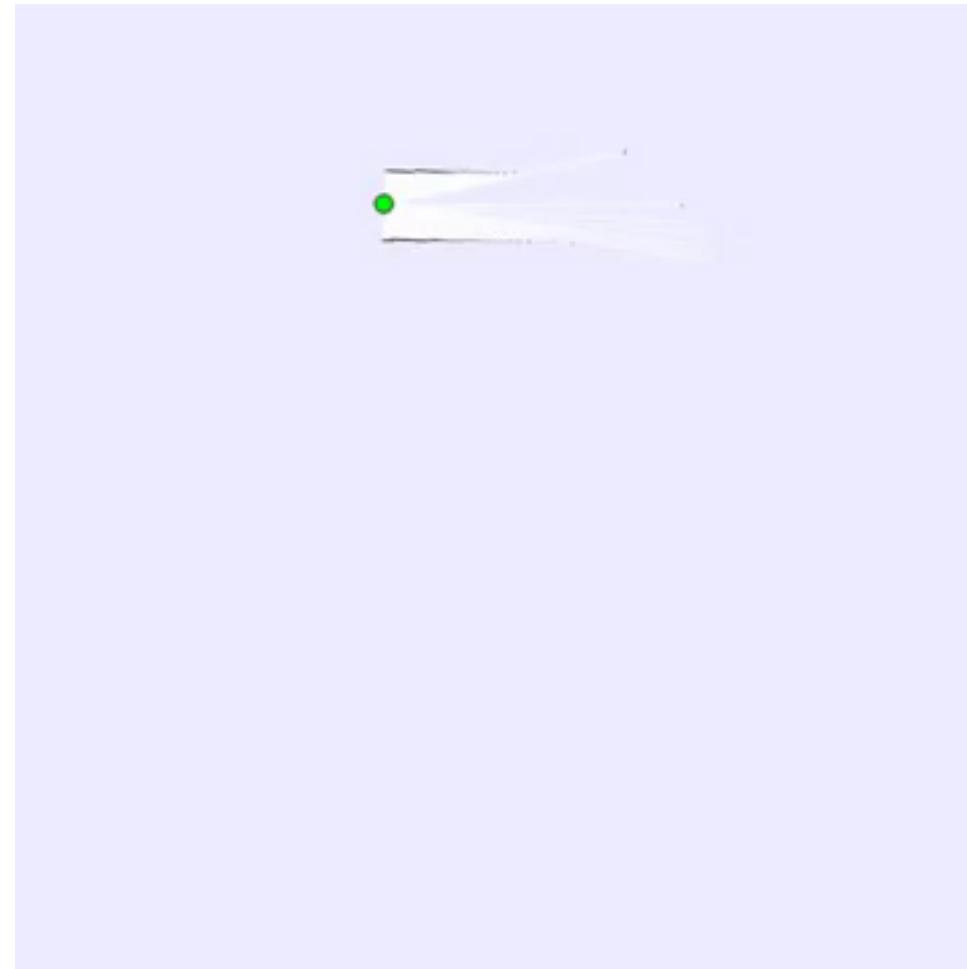


- Scan matching (E. Olson, 2009)



Example

The map is obtained by transforming and accumulating laser scans using robot poses solved by pose graph SLAM



2D Pose Graph SLAM

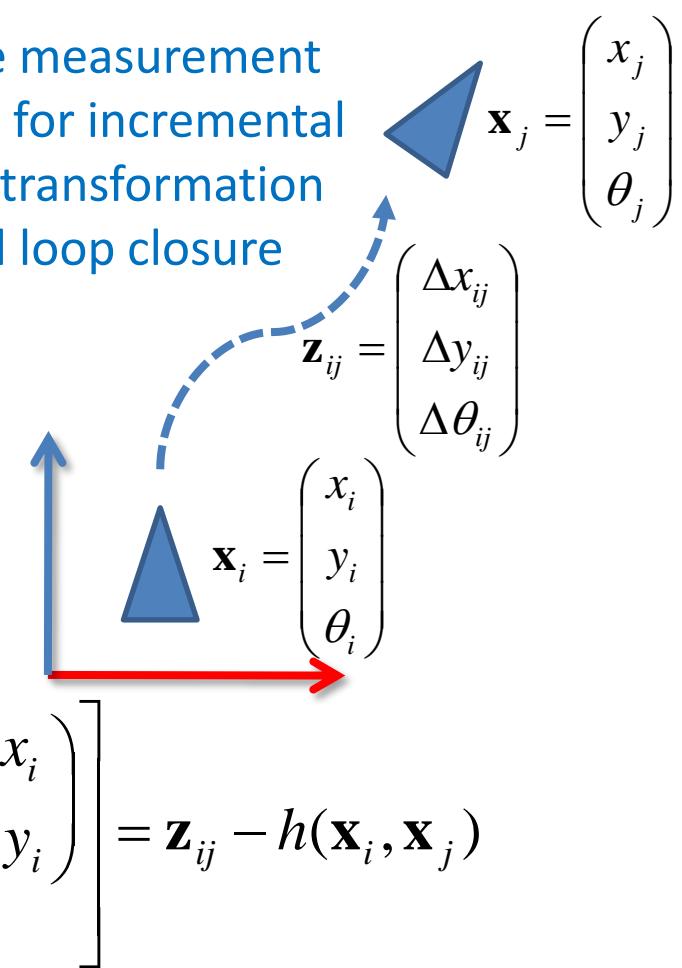
$$\begin{aligned}
 \hat{\mathbf{x}}_{1:t} &= \arg \max_{\mathbf{x}_{1:t}} p(\mathbf{x}_{1:t} | \mathbf{z}_{1:N}) \\
 &= \arg \max_{\mathbf{x}_{1:t}} \prod_{i,j} p(\mathbf{z}_{ij} | \mathbf{x}_{1:t}) \cdot p(\mathbf{x}_1) \\
 &= \arg \min_{\mathbf{x}_{1:t}} \sum_{i,j} \boxed{\mathbf{r}_{z_{ij}}^T \Sigma_{z_{ij}}^{-1} \mathbf{r}_{z_{ij}}} + \mathbf{r}_{x_1}^T \Sigma_{x_1}^{-1} \mathbf{r}_{x_1}
 \end{aligned}$$

Measurement Residual

$$\mathbf{r}_{z_{ij}} = \begin{pmatrix} \Delta x_{ij} \\ \Delta y_{ij} \\ \Delta \theta_{ij} \end{pmatrix} - \left[\begin{pmatrix} \cos \theta_i & -\sin \theta_i \\ \sin \theta_i & \cos \theta_i \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x_j - x_i \\ y_j - y_i \\ \theta_j - \theta_i \end{pmatrix} \right] = \mathbf{z}_{ij} - h(\mathbf{x}_i, \mathbf{x}_j)$$

$$\mathbf{r}_{z_{ij}} \sim N(0, \Sigma_{z_{ij}})$$

Same measurement model for incremental pose transformation and loop closure



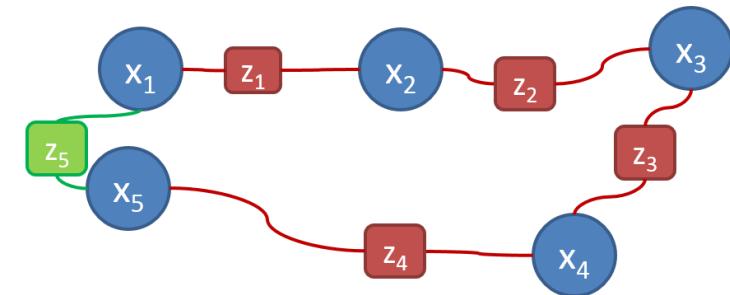
SLAM as Nonlinear Least Square

- Nonlinear least square

$$\hat{\mathbf{X}}_{1:t} = \arg \min_{\mathbf{X}_{1:t}} \sum_{i,j} \mathbf{r}_{z_{ij}}^T \Sigma_{z_{ij}}^{-1} \mathbf{r}_{z_{ij}} + \cancel{\mathbf{r}_{x_1}^T \Sigma_{x_1}^{-1} \mathbf{r}_{x_1}}$$

- Linearization

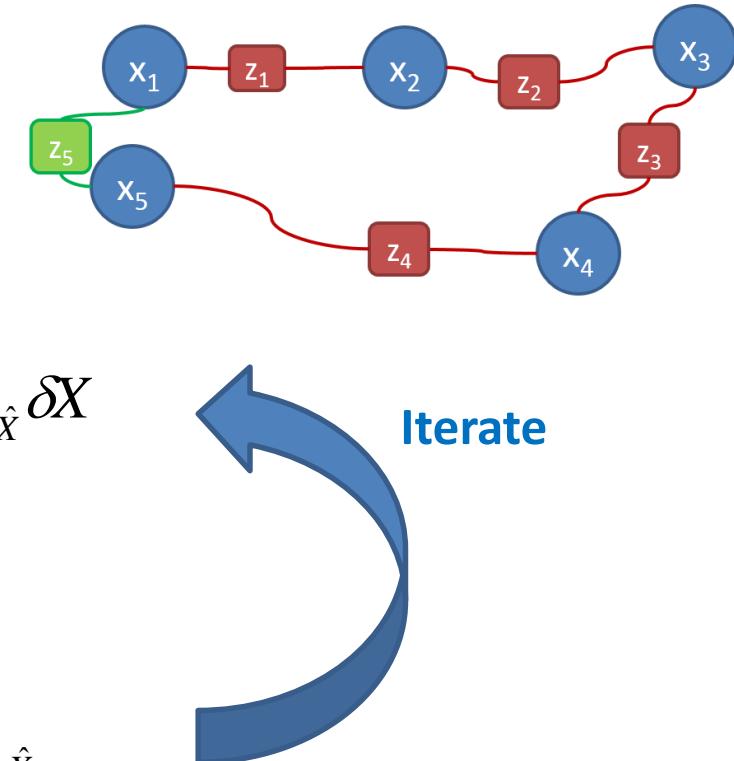
$$\mathbf{r}_{z_{ij}} \approx \mathbf{r}_{z_{ij}, \hat{X}} + \frac{\partial \mathbf{r}_{z_{ij}, \hat{X}}}{\partial X} \Big|_{\hat{X}} \delta X = \mathbf{r}_{z_{ij}, \hat{X}} + \mathbf{J}_{z_{ij}, \hat{X}} \delta X$$



SLAM as Nonlinear Least Square

- Nonlinear least square

$$\hat{\mathbf{x}}_{1:t} = \arg \min_{\mathbf{x}_{1:t}} \sum_{i,j} \mathbf{r}_{z_{ij}}^T \Sigma_{z_{ij}}^{-1} \mathbf{r}_{z_{ij}} + \cancel{\mathbf{r}_{x_1}^T \Sigma_{x_1}^{-1} \mathbf{r}_{x_1}}$$



- Linearization

$$\mathbf{r}_{z_{ij}} \approx \mathbf{r}_{z_{ij}, \hat{X}} + \frac{\partial \mathbf{r}_{z_{ij}, \hat{X}}}{\partial X} \Big|_{\hat{X}} \delta X = \mathbf{r}_{z_{ij}, \hat{X}} + \mathbf{J}_{z_{ij}, \hat{X}} \delta X$$

- Gauss-Newton method

$$\sum_{i,j} \mathbf{J}_{z_{ij}, \hat{X}}^T \Sigma_{z_{ij}}^{-1} \mathbf{J}_{z_{ij}, \hat{X}} \cdot \delta X = - \sum_{i,j} \mathbf{J}_{z_{ij}, \hat{X}}^T \Sigma_{z_{ij}}^{-1} \mathbf{r}_{z_{ij}, \hat{X}}$$

$$\hat{X}^{(k+1)} = \hat{X}^{(k)} + \delta X$$

More Example

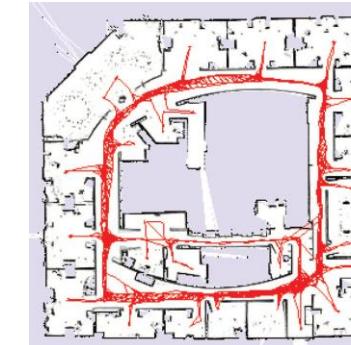
The map is obtained by transforming and accumulating laser scans using robot poses solved by pose graph SLAM



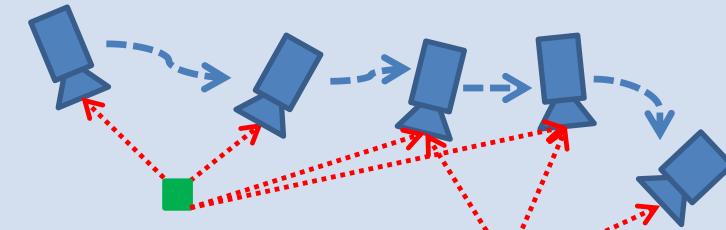
Outline

$$P(x \mid y, z) = \frac{P(y \mid x, z) P(x \mid z)}{P(y \mid z)}$$

- The Basics
- 2D Pose Graph SLAM

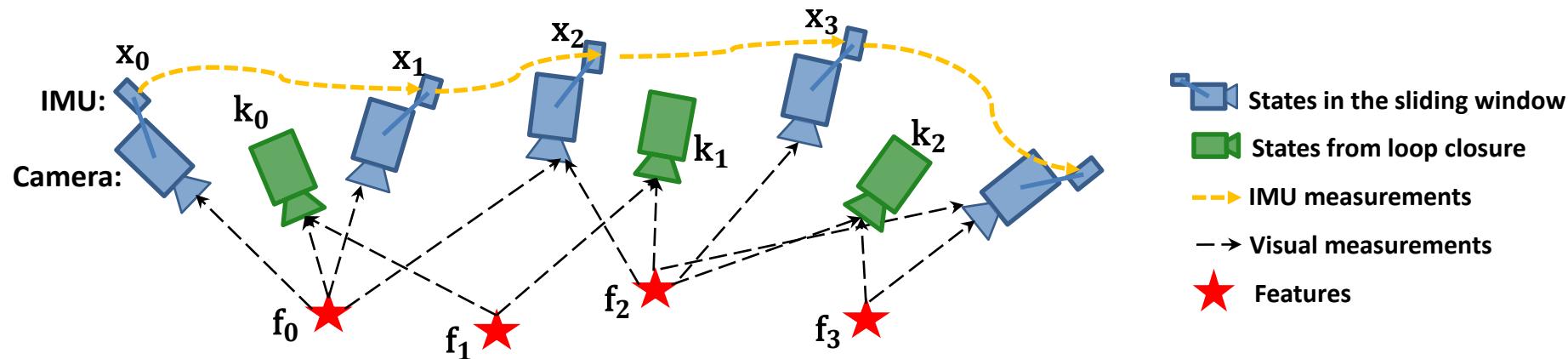
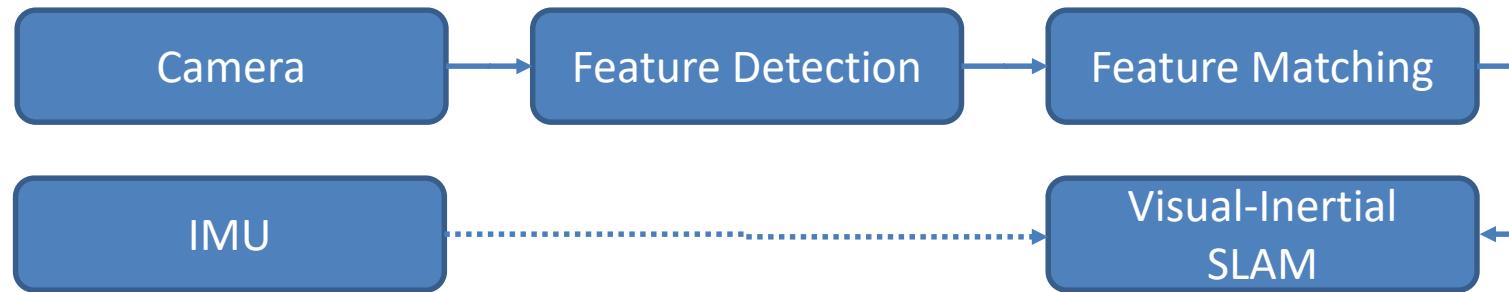


- Monocular Visual-Inertial SLAM



Visual-Inertial SLAM Pipeline

- A SLAM system that use multiple sensors
- Still a graphical representation, with different types of edges



Camera Setup

- Monocular
 - Depth Unknown



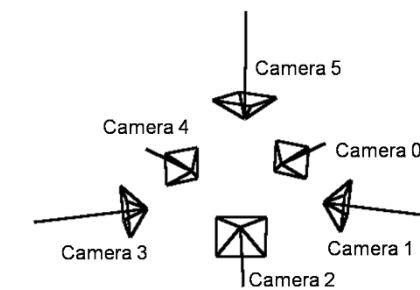
- Stereo
 - Able to compute depth
 - Depth accuracy affect by baseline, resolution, and calibration



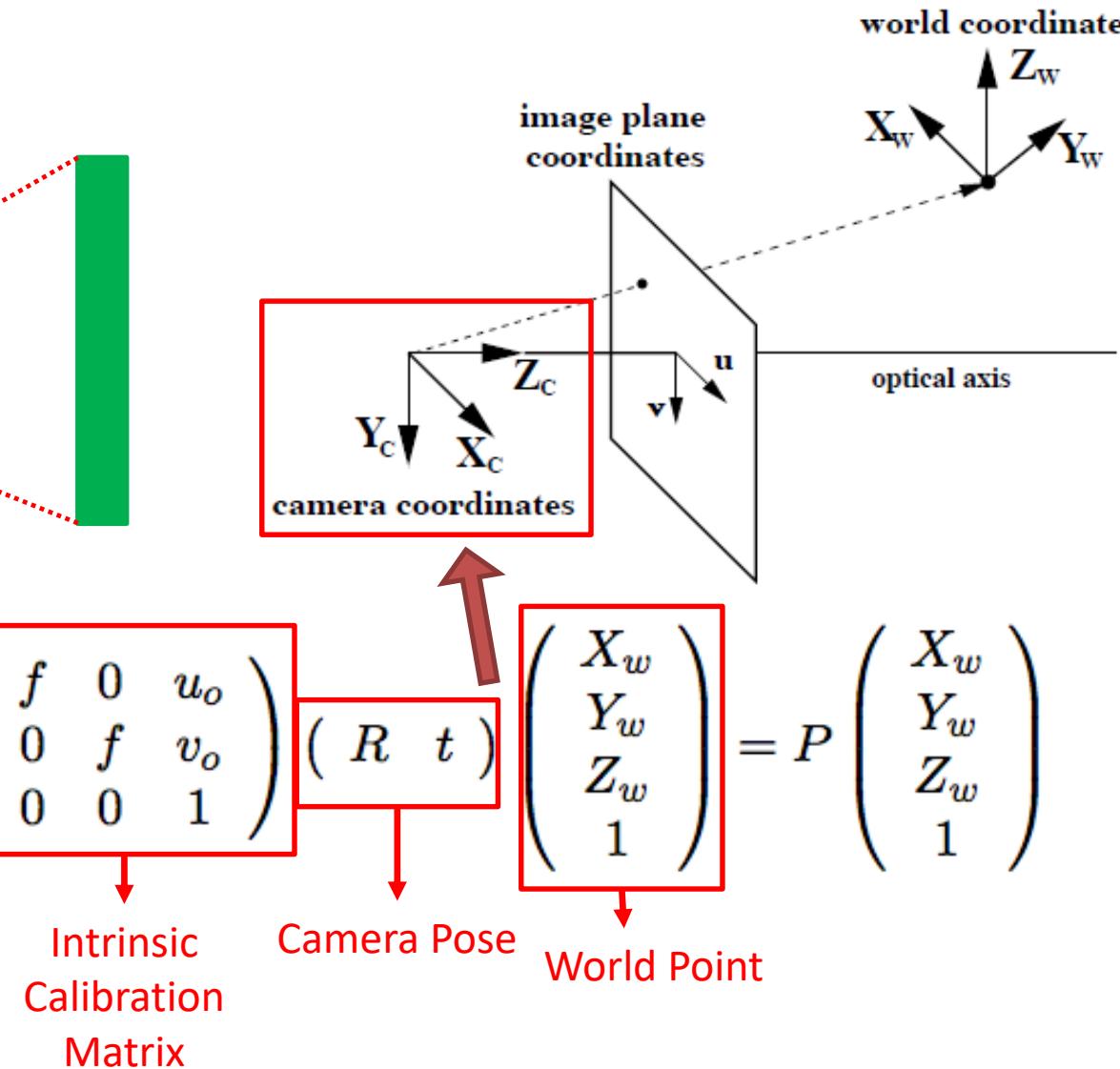
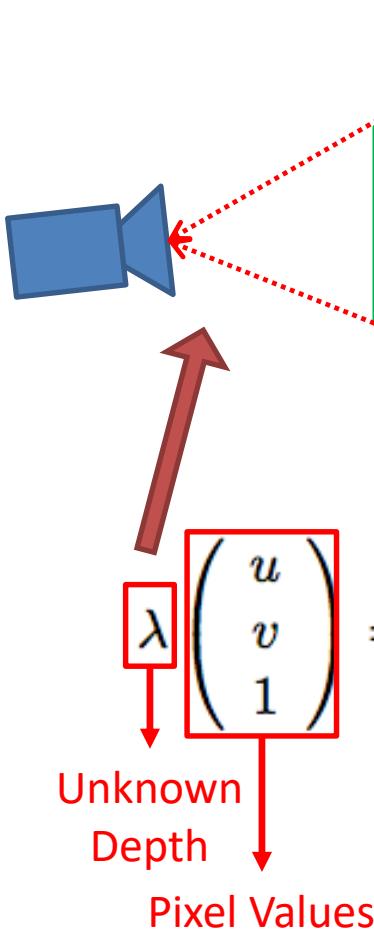
- Multi-Camera
 - Overlapping / Non-overlapping field-of-view



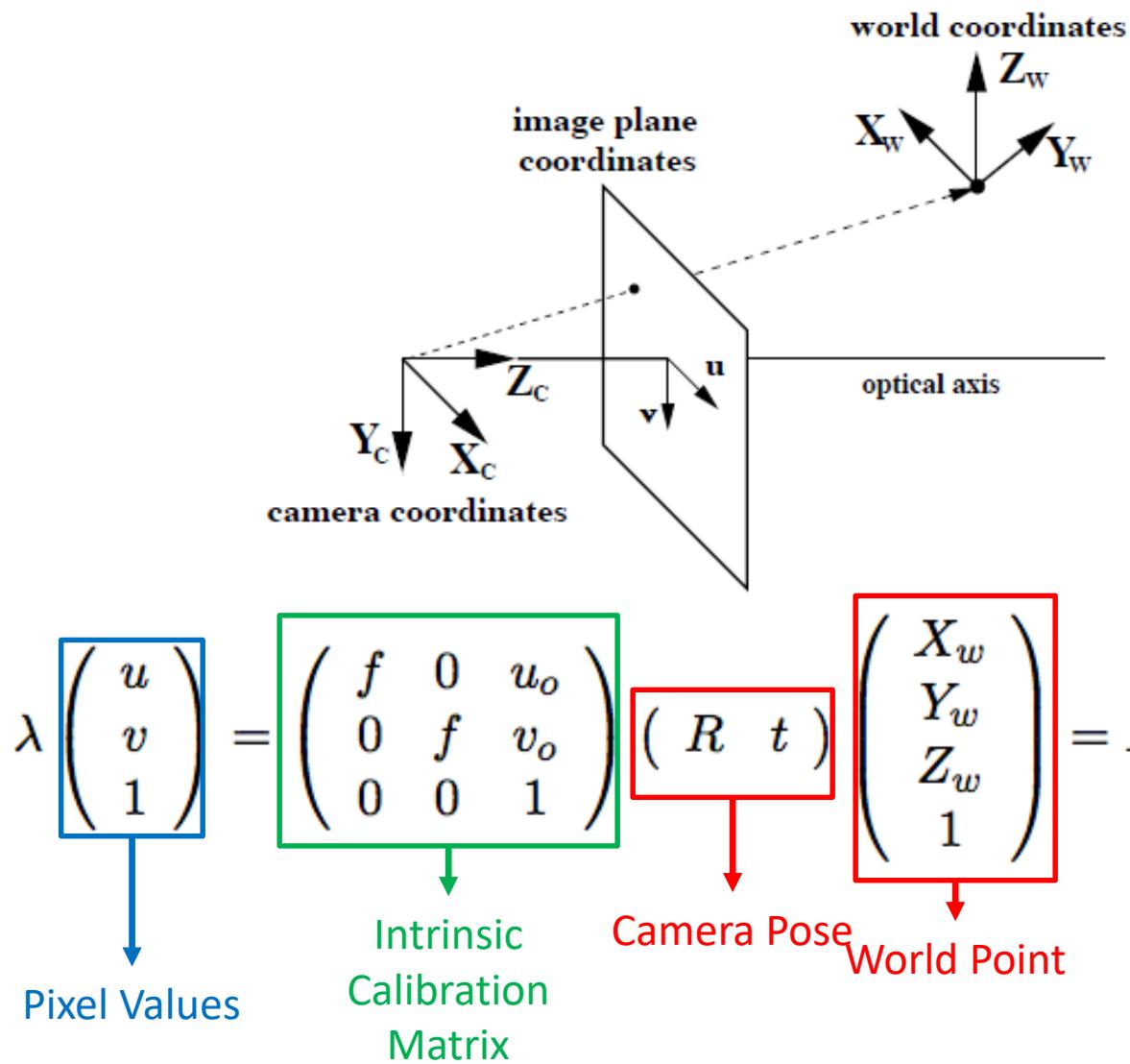
Omnidirectional multi-camera system
Ladybug



Pin-hole Camera Model



Problem Formulation



Why Monocular?

- Minimum structural requirements
- Widely available sensors
- Applications:
 - State estimation for small drones
 - Mobile augmented reality



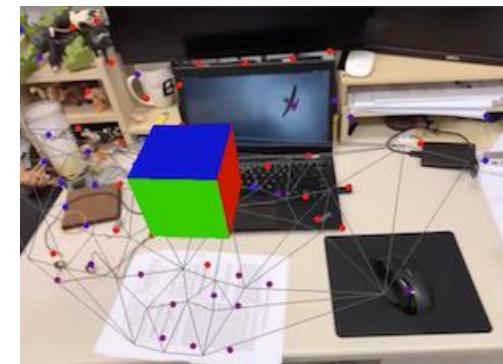
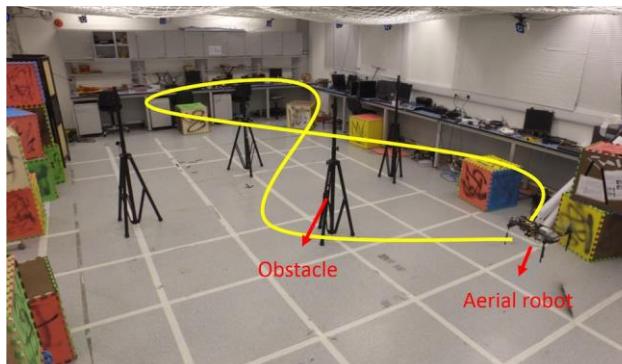
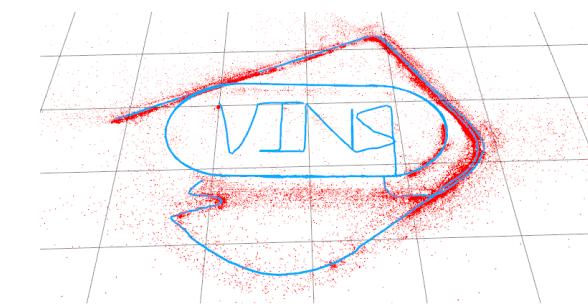
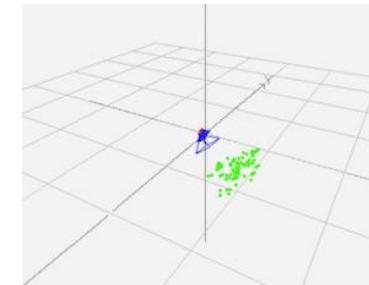
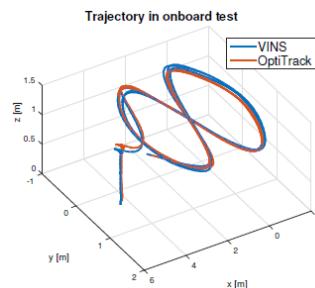
Why IMU?

- IMU measures:
 - Linear acceleration
 - Angular velocity
- Pros:
 - Almost always available and outlier-free
 - Very high-rate measurements
 - Very mature technology, widely available at very low cost
 - Remarkable performance improvement during aggressive motions
 - **Metric scale measurement**
- Cons:
 - Noisy sensor, cannot double integrate to obtain position
 - Synchronization and inter-sensor calibration requirements
 - Observability and numerical stability issues
 - Unable to operate when inertial and visual measurements are not in the same frame (e.g. on cars or trains)



Requirements

- Metric scale estimation using only one camera
- Mostly for **state estimation** (localization), map is sparse
- Robust and smooth odometry – **local accuracy**
- Loop closure – **global consistency**



Related work

- MSC-KF (Mourikis and Roumeliotis, 2007)
- 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
- Google ARCore

Our Solution: VINS-Mono

HKUST-Aerial-Robotics / VINS-Mono

Unwatch 209 Unstar 2.3k Fork 1.3k

Code Issues 198 Pull requests 1 Actions Projects 0 Wiki Security 0 Insights Settings

A Robust and Versatile Monocular Visual-Inertial State Estimator Edit

state-estimation vio vins Manage topics

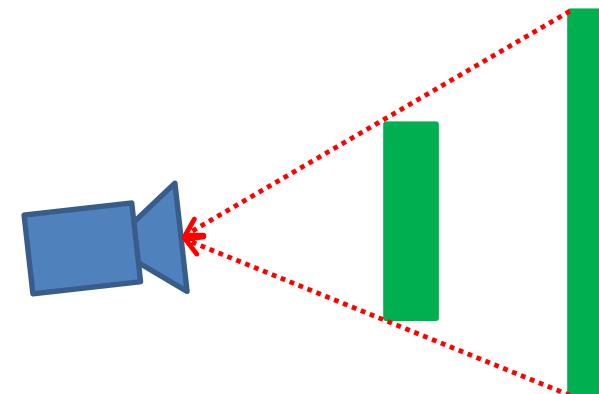
76 commits 3 branches 0 packages 0 releases 4 contributors GPL-3.0

Branch: master New pull request Create new file Upload files Find file Clone or download

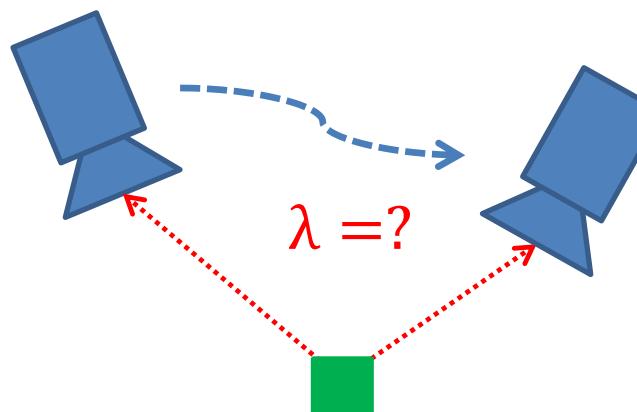
shaozu	update readme	Latest commit dc62ec4 on Mar 21, 2019
ar_demo	fix ar_demo image interface	15 months ago
benchmark_publisher	another warning	2 years ago
camera_model	add Eigen3 cmake	2 years ago
config	Merge branch 'master' of github.com:HKUST-Aerial-Robotics/VINS-Mono	14 months ago
docker	update docker	14 months ago

Challenges: Monocular Vision

- Scale ambiguity

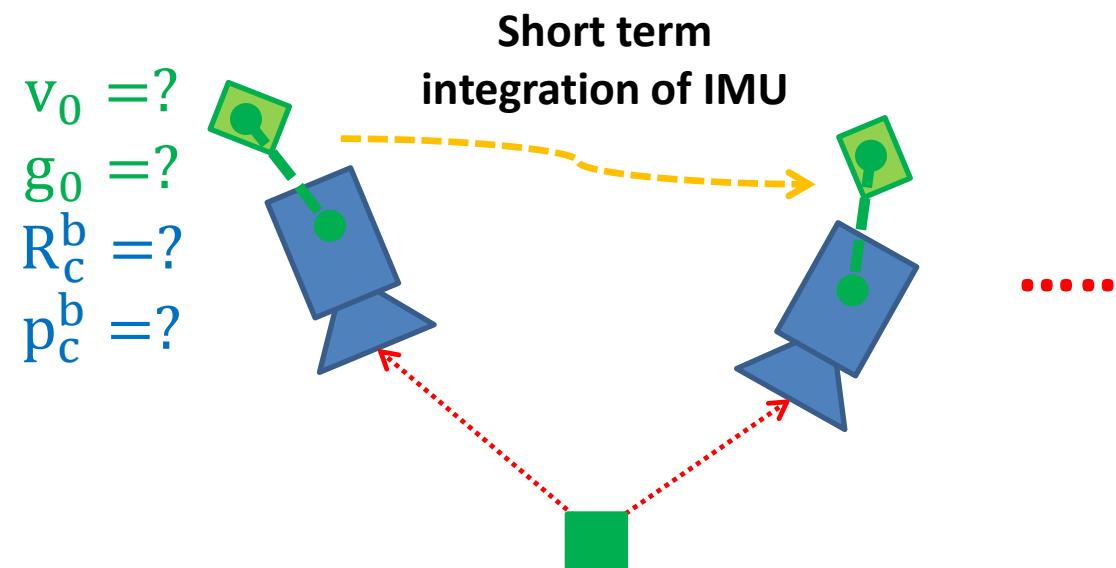


- Up-to-scale motion estimation and 3D reconstruction
(Structure from Motion)



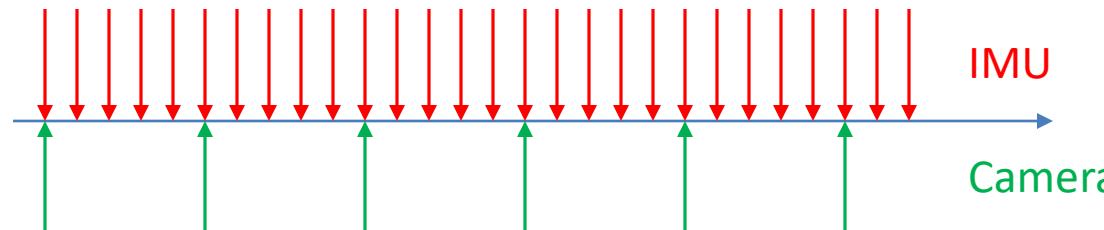
Challenges: Monocular Visual-Inertial Systems

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

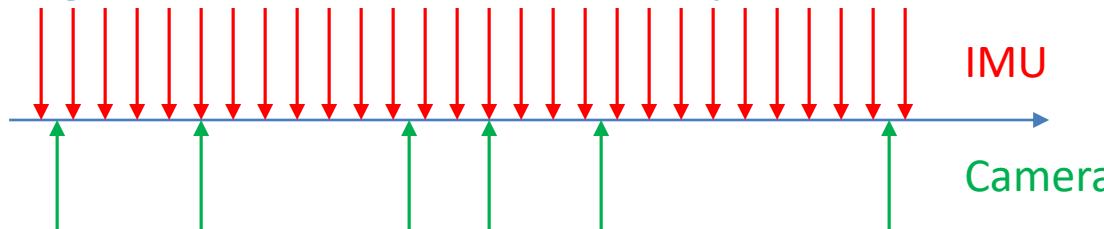


Challenges: Synchronization

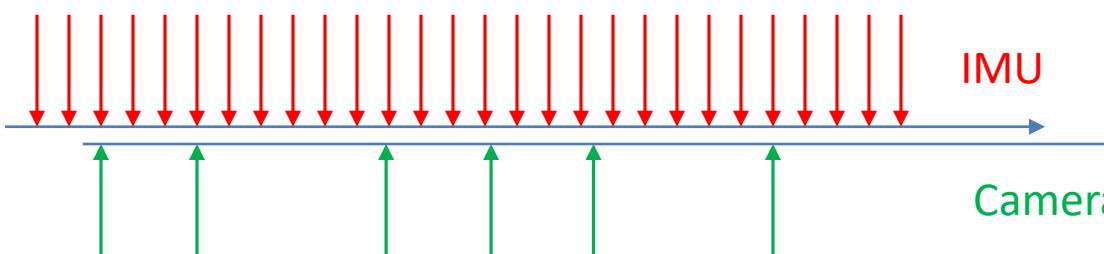
- Best: Sensors are hardware-triggered



- OK: Sensors have the same clock (e.g. running on the same system clock or have global clock correction) but capture data at different times

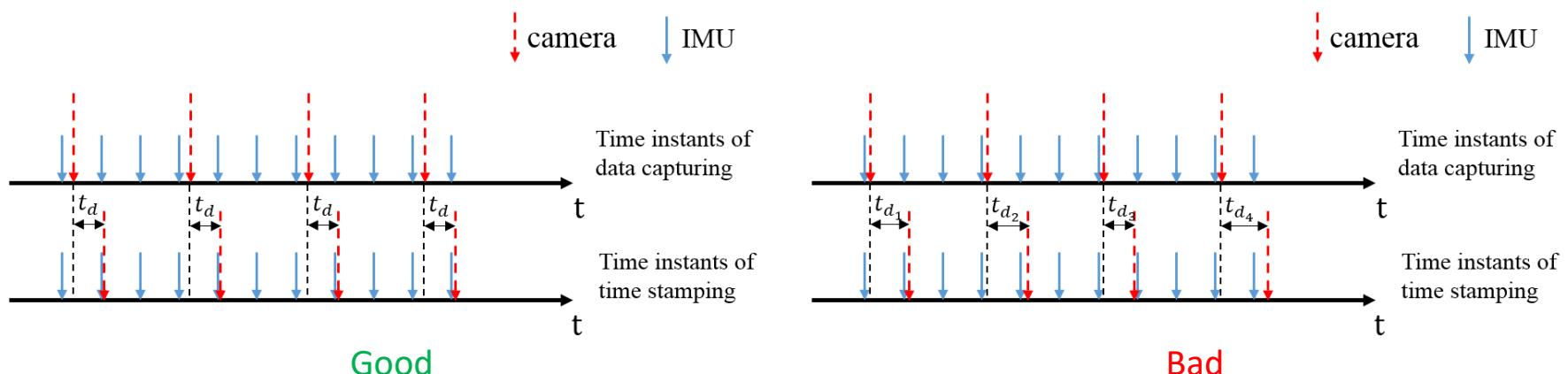


- Bad: Sensors have different clocks (e.g. each sensor has its own oscillator)



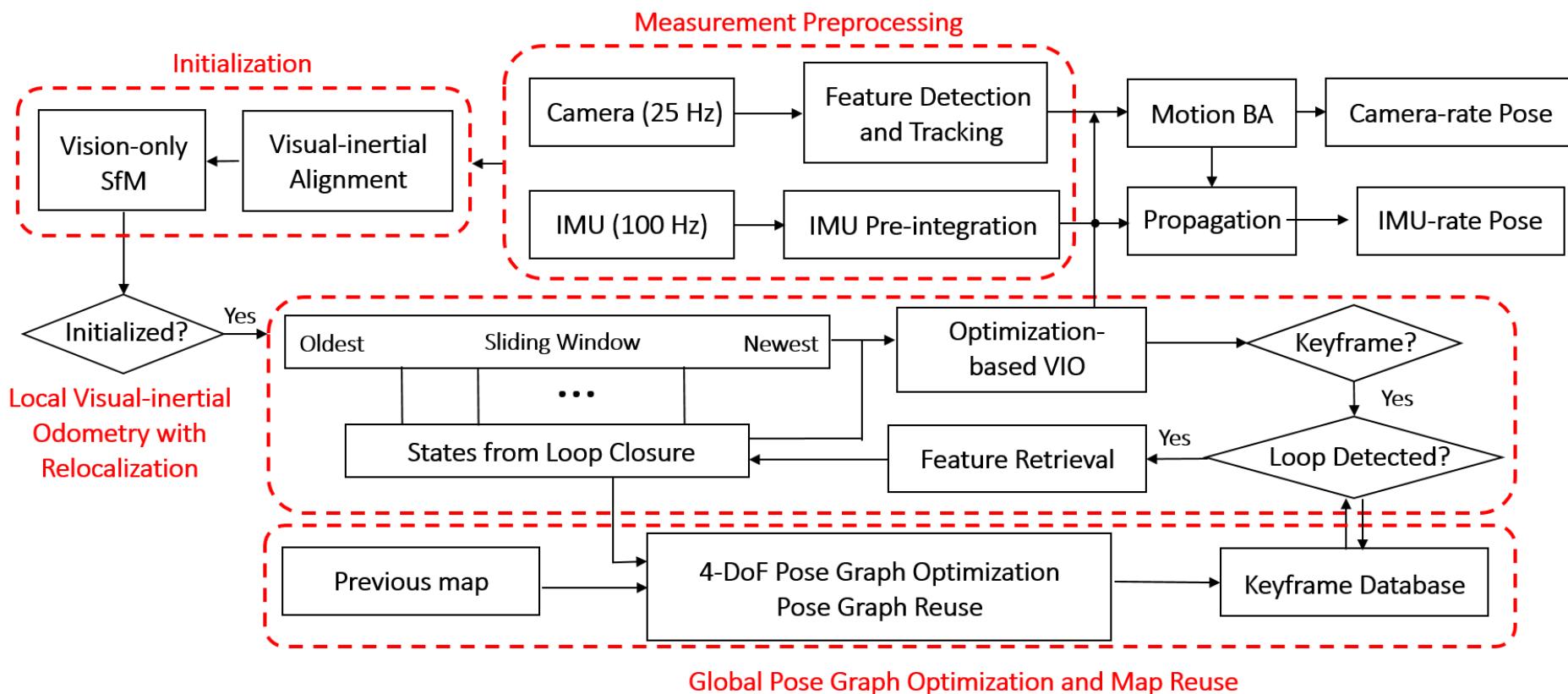
Challenges: Timestamps

- **Timestamp:** how the time for each sensor measurement is tagged
- **Best:** timestamping is done at data capture
- **OK:** fixed latency for time stamping
 - e.g. time is tagged on low-level hardware after some fixed-duration data processing, and will not be affected by any dynamic OS scheduling tasks
- **Bad:** variable latency in time stamping
 - e.g. plug two sensors into USB ports and time stamp according to the PC time. Time stamping is affected by data transmission latency from the sensor to PC



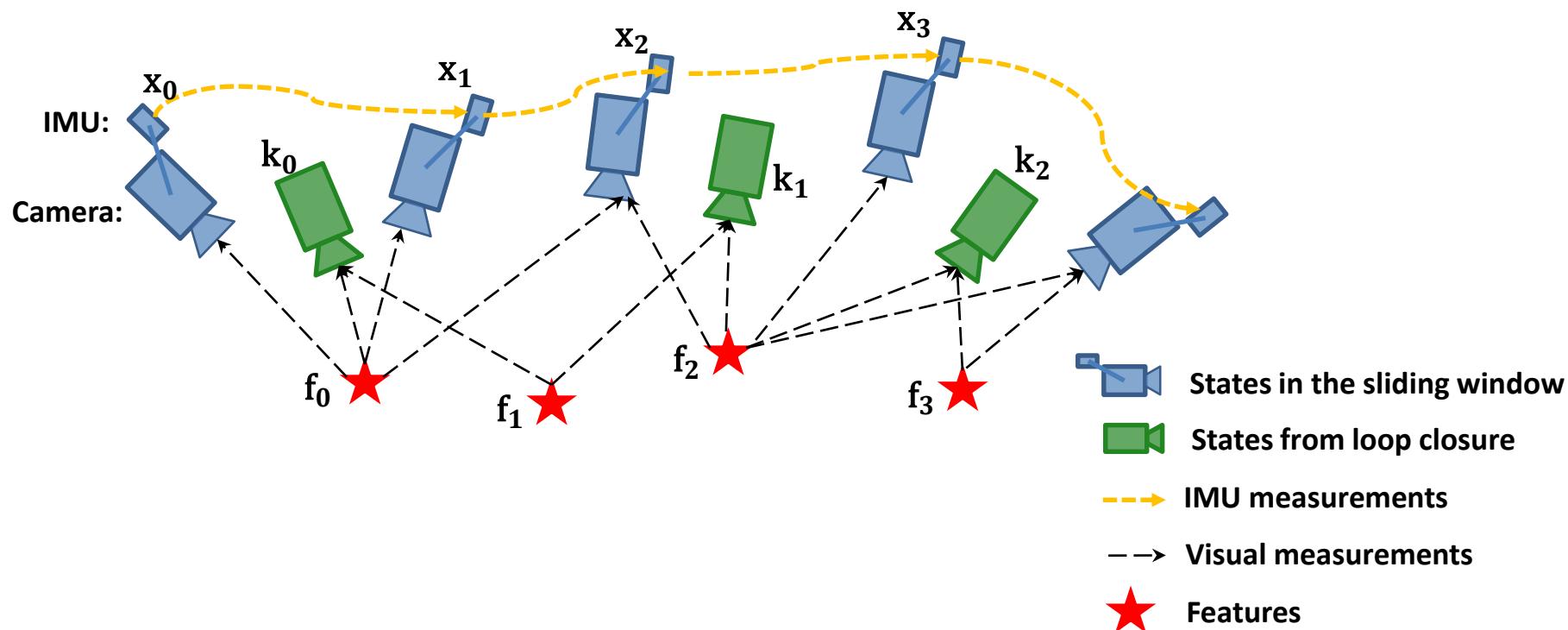
Monocular Visual-Inertial SLAM

- System diagram



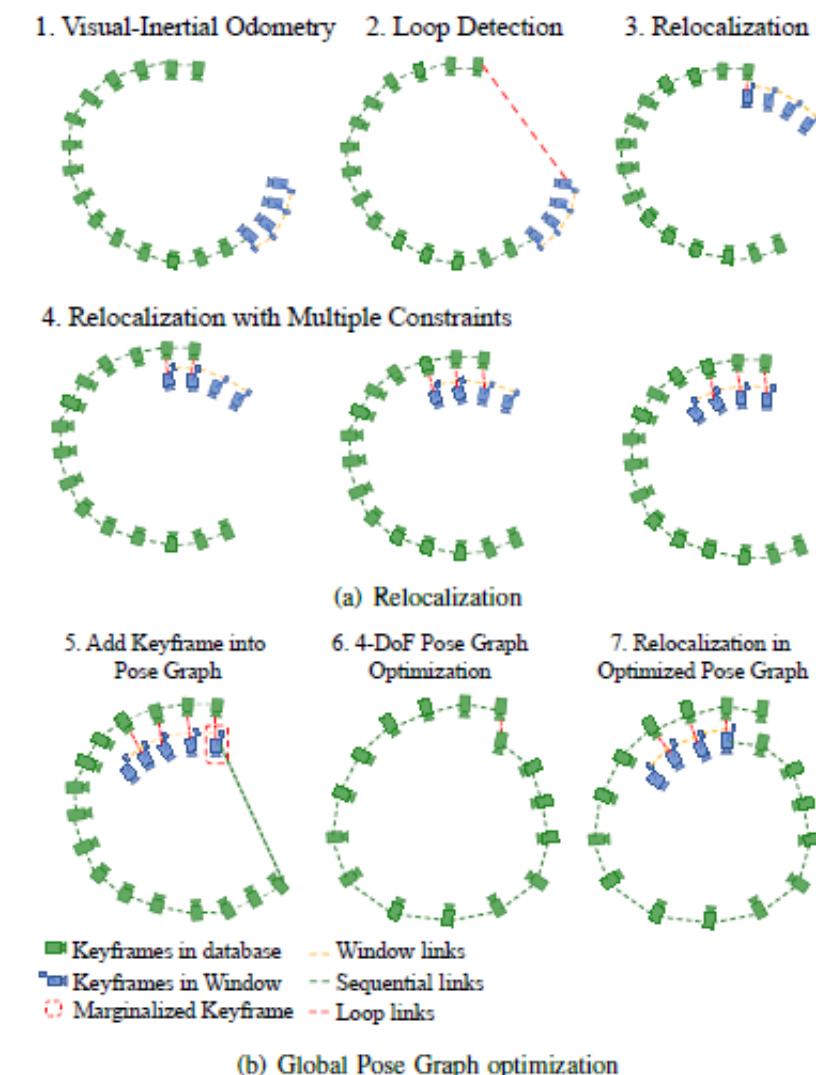
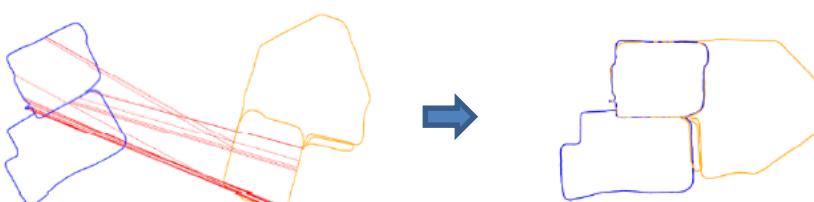
Monocular Visual-Inertial SLAM

- Monocular visual-inertial odometry with relocalization
 - For local accuracy
 - Achieved via *sliding window* visual-inertial **bundle adjustment**



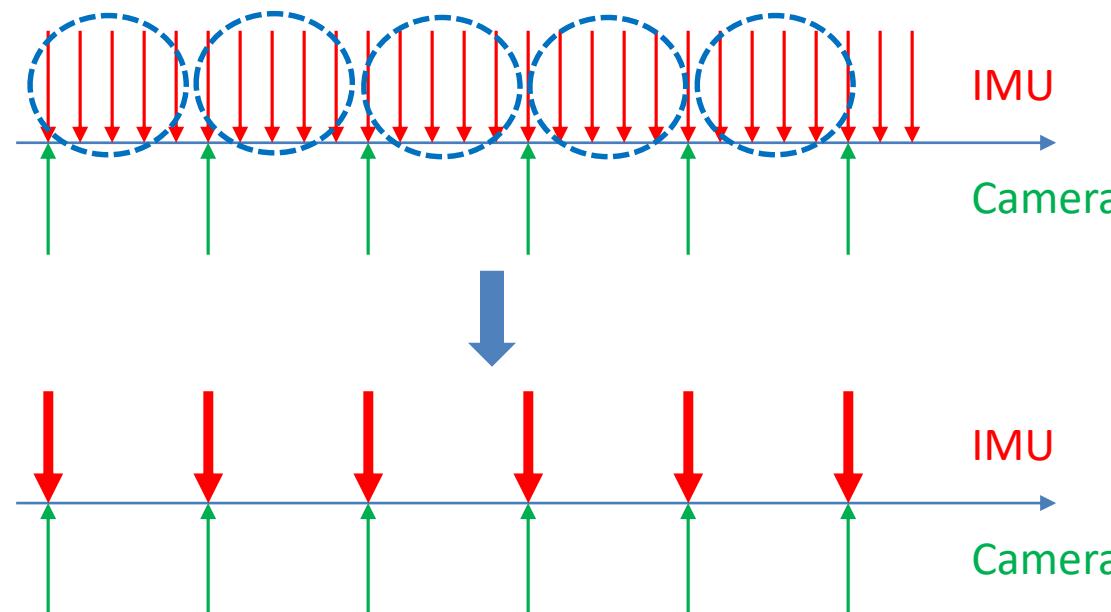
Monocular Visual-Inertial SLAM

- Global pose graph SLAM
 - For global consistency
 - Fully integrated with tightly-coupled re-localization
- Map reuse
 - Save map at any time
 - Load map and re-localize with respect to it
 - Pose graph merging



How to Use IMU?

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



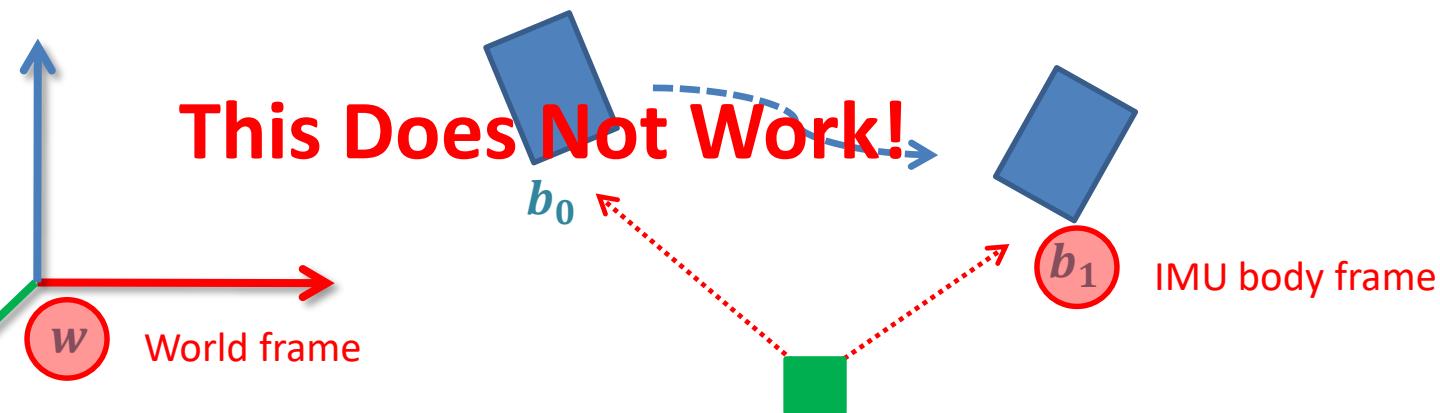
The Bad of IMU Integration in the Global Frame

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

$$\mathbf{p}_{b_{k+1}}^w = \mathbf{p}_{b_k}^w + \mathbf{v}_{b_k}^w \Delta t_k + \iint_{t \in [t_k, t_{k+1}]} (\mathbf{R}_t^w (\hat{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a) - \mathbf{g}^w) dt^2$$

$$\mathbf{v}_{b_{k+1}}^w = \mathbf{v}_{b_k}^w + \int_{t \in [t_k, t_{k+1}]} (\mathbf{R}_t^w \hat{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a) - \mathbf{g}^w dt$$

$$\mathbf{q}_{b_{k+1}}^w = \mathbf{q}_{b_k}^w \otimes \int_{t \in [t_k, t_{k+1}]} \frac{1}{2} \boldsymbol{\Omega} (\hat{\boldsymbol{\omega}}_t - \mathbf{b}_{w_t} - \mathbf{n}_w) \mathbf{q}_t^{b_k} dt, \quad \boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -[\boldsymbol{\omega}]_\times & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix}, [\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}.$$



IMU 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

$$\mathbf{R}_w^{b_k} \mathbf{p}_{b_{k+1}}^w = \mathbf{R}_w^{b_k} (\mathbf{p}_{b_k}^w + \mathbf{v}_{b_k}^w \Delta t_k - \frac{1}{2} \mathbf{g}^w \Delta t_k^2) + \alpha_{b_{k+1}}^{b_k}$$

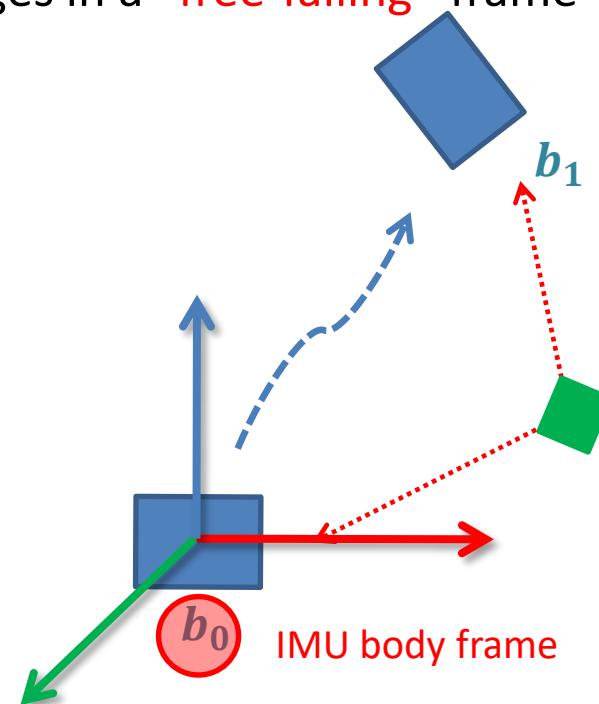
$$\mathbf{R}_w^{b_k} \mathbf{v}_{b_{k+1}}^w = \mathbf{R}_w^{b_k} (\mathbf{v}_{b_k}^w - \mathbf{g}^w \Delta t_k) + \beta_{b_{k+1}}^{b_k}$$

$$\mathbf{q}_w^{b_k} \otimes \mathbf{q}_{b_{k+1}}^w = \gamma_{b_{k+1}}^{b_k},$$

$$\alpha_{b_{k+1}}^{b_k} = \iint_{t \in [t_k, t_{k+1}]} \mathbf{R}_t^{b_k} (\hat{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a) dt^2$$

$$\beta_{b_{k+1}}^{b_k} = \int_{t \in [t_k, t_{k+1}]} \mathbf{R}_t^{b_k} (\hat{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a) dt$$

$$\gamma_{b_{k+1}}^{b_k} = \int_{t \in [t_k, t_{k+1}]} \frac{1}{2} \Omega (\hat{\omega}_t - \mathbf{b}_{w_t} - \mathbf{n}_w) \gamma_t^{b_k} dt.$$



IMU Pre-Integration on Manifold

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

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

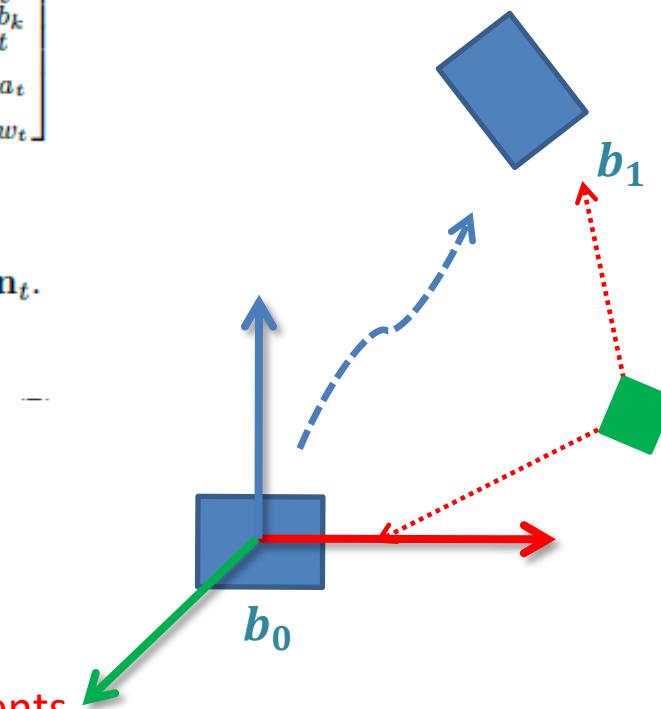
Bias uncertainty

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

- Discrete-time implementation

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

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

$$\mathbf{J}_{b_k} = \mathbf{I},$$

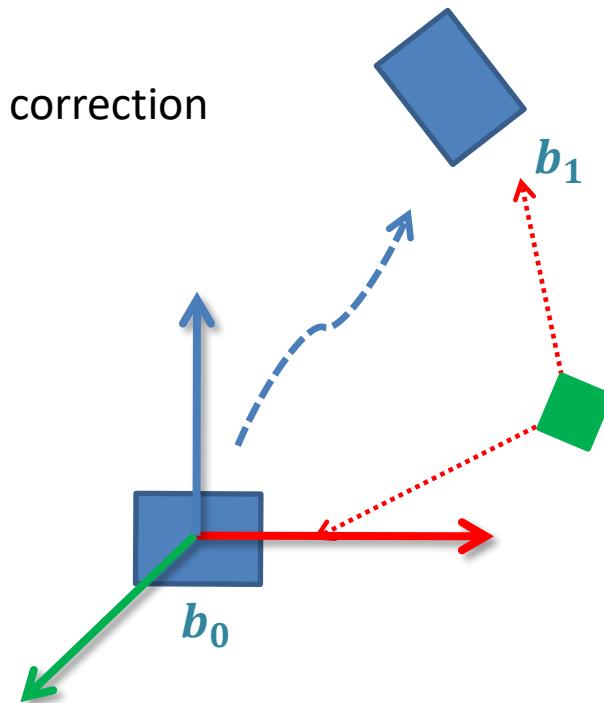
$$\mathbf{J}_{t+\delta t} = (\mathbf{I} + \mathbf{F}_t \delta t) \mathbf{J}_t, \quad t \in [k, k+1]$$

- And write down the linearized model for bias correction

$$\alpha_{b_{k+1}}^{b_k} \approx \hat{\alpha}_{b_{k+1}}^{b_k} + \mathbf{J}_{b_a}^\alpha \delta \mathbf{b}_{a_k} + \mathbf{J}_{b_w}^\alpha \delta \mathbf{b}_{w_k}$$

$$\beta_{b_{k+1}}^{b_k} \approx \hat{\beta}_{b_{k+1}}^{b_k} + \mathbf{J}_{b_a}^\beta \delta \mathbf{b}_{a_k} + \mathbf{J}_{b_w}^\beta \delta \mathbf{b}_{w_k}$$

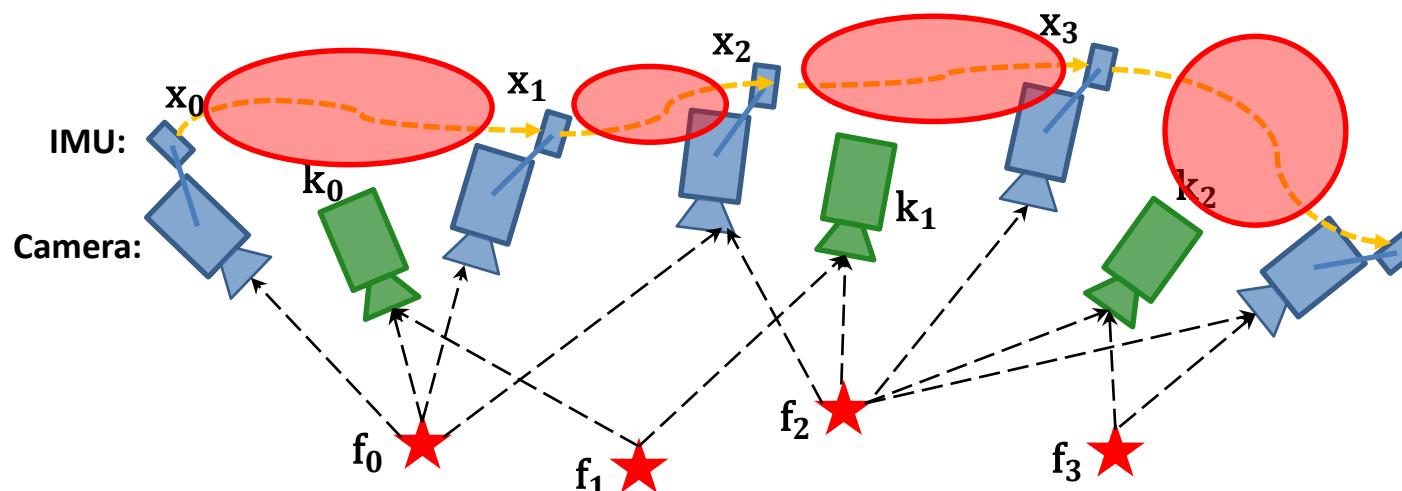
$$\gamma_{b_{k+1}}^{b_k} \approx \hat{\gamma}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \mathbf{J}_{b_w}^\gamma \delta \mathbf{b}_{w_k} \end{bmatrix}$$



IMU Pre-Integration on Manifold

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

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

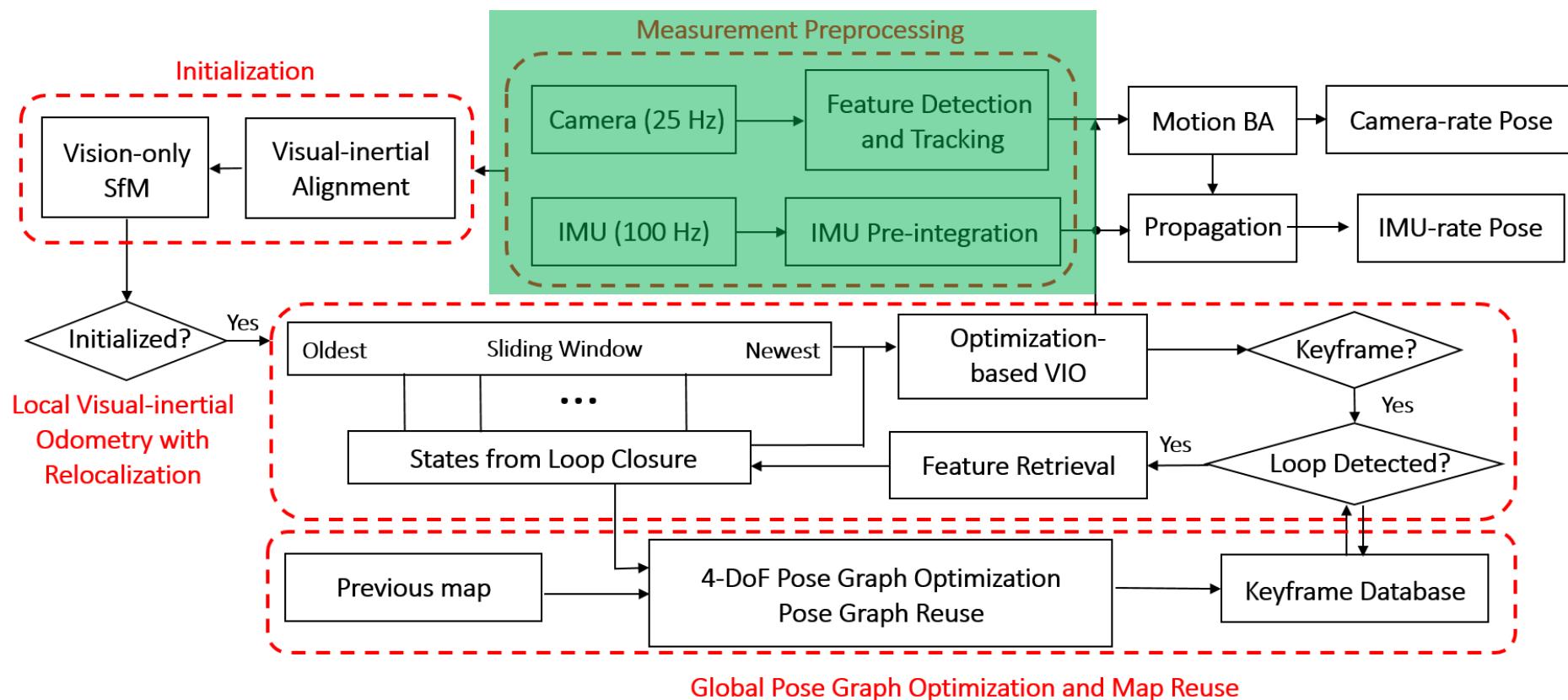


Vision Front-End

- Simple feature processing pipeline
 - Harris corners...
 - KLT tracker...
 - Track between consecutive frames
 - RANSAC for preliminary outlier removal
- Keyframe selection
 - Case 1: Rotation-compensated average feature parallax is larger than a threshold
 - Avoid numerical issues caused by poorly triangulated features
 - Case 2: Number of tracked features in the current frame is less than a threshold
 - Avoid losing tracking
 - All frames are used for optimization, but non-keyframes are removed first

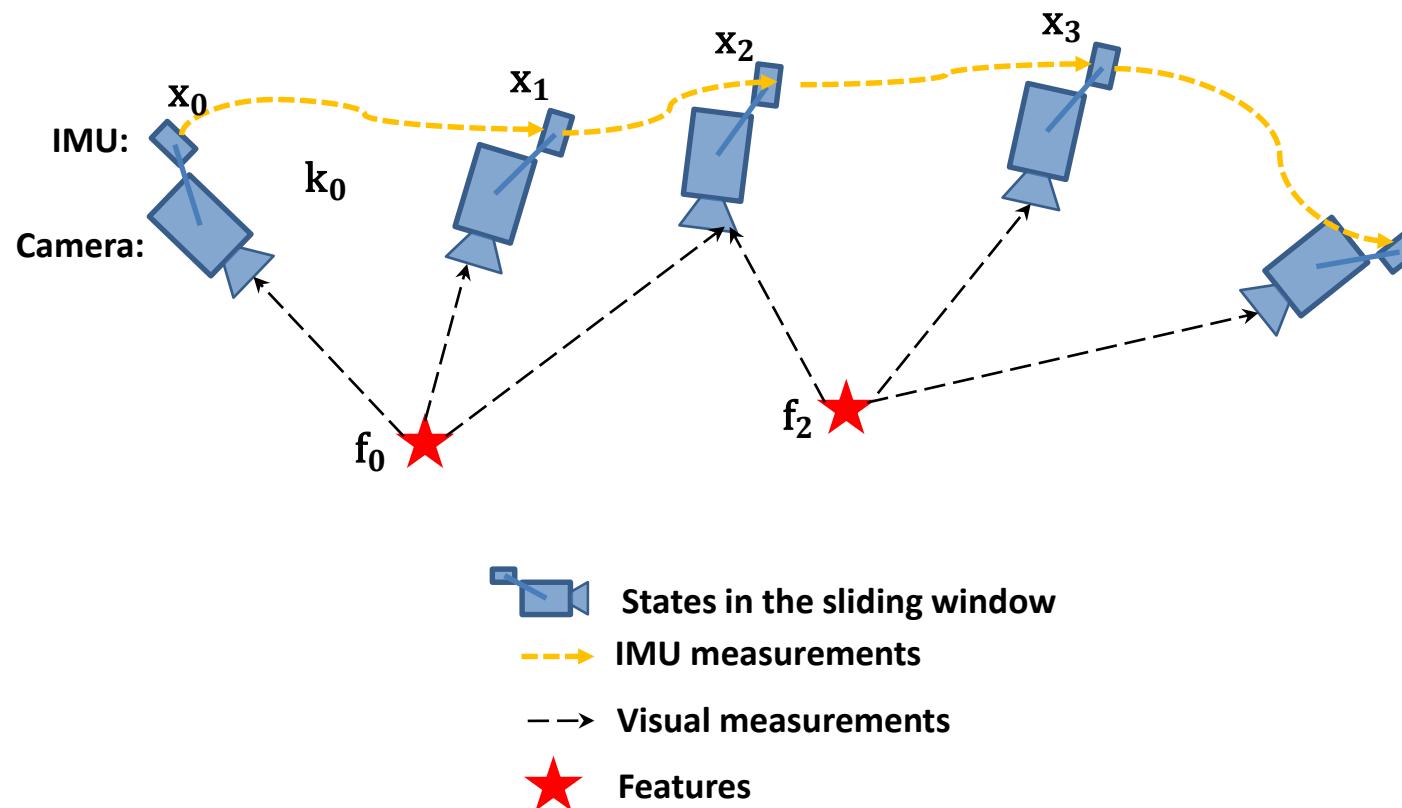
Monocular Visual-Inertial SLAM

- 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 extrinsic calibration** simultaneously:

$$\mathcal{X} = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{x}_c^b, \lambda_0, \lambda_1, \dots, \lambda_m]$$

$$\mathbf{x}_k = [\mathbf{p}_{b_k}^w, \mathbf{v}_{b_k}^w, \mathbf{q}_{b_k}^w, \mathbf{b}_a, \mathbf{b}_g], k \in [0, n]$$

$$\mathbf{x}_c^b = [\mathbf{p}_c^b, \mathbf{q}_c^b],$$

- Minimize residuals from all sensors

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

IMU measurement residual Vision measurement residual

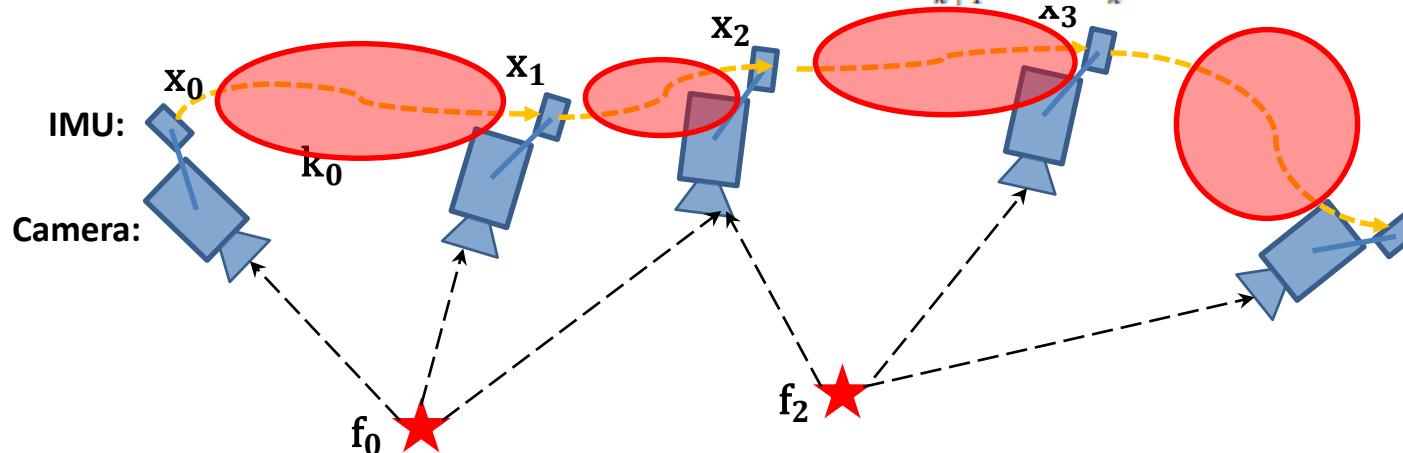
Prior from marginalization Covariance from IMU pre-integration Pixel reprojection covariance

Monocular Visual-Inertial Odometry

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

IMU pre-integration “blocks”

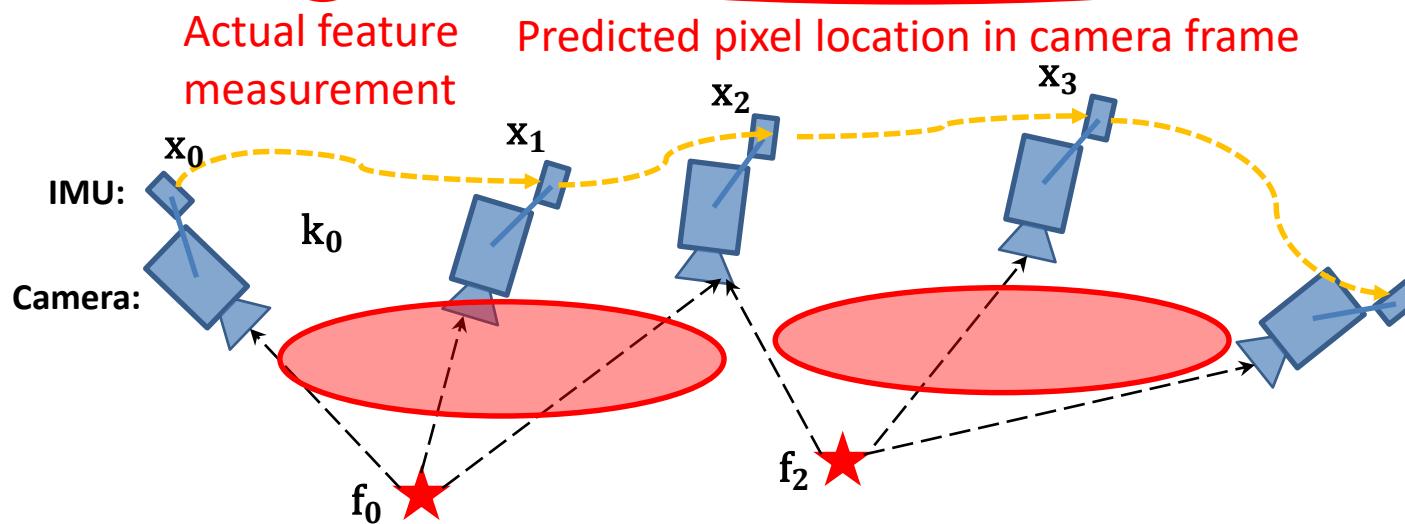
$$\mathbf{r}_B(\hat{\mathbf{z}}_{b_{k+1}}^{b_k}, \mathcal{X}) = \begin{bmatrix} \delta\alpha_{b_{k+1}}^{b_k} \\ \delta\beta_{b_{k+1}}^{b_k} \\ \delta\theta_{b_{k+1}}^{b_k} \\ \delta\mathbf{b}_a \\ \delta\mathbf{b}_g \end{bmatrix} = \begin{bmatrix} \mathbf{R}_w^{b_k} (\mathbf{p}_{b_{k+1}}^w - \mathbf{p}_{b_k}^w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



Monocular Visual-Inertial Odometry

- Vision measurement residual
 - Pixel reprojection error
 - Inverse depth model, at least 2 observations per feature, first observation to define feature direction

$$\mathbf{r}_c(\hat{\mathbf{z}}_l^{c_j}, \mathcal{X}) = \begin{bmatrix} \hat{u}_l^{c_j} \\ \hat{v}_l^{c_j} \end{bmatrix} - \pi_c(\mathbf{R}_b^c(\mathbf{R}_w^b(\mathbf{R}_{b_t}^w(\mathbf{R}_c^b \frac{1}{\lambda_l} \pi_c^{-1} \begin{bmatrix} u_l^{c_t} \\ v_l^{c_t} \end{bmatrix} + \mathbf{p}_c^b) + \mathbf{p}_{b_t}^w - \mathbf{p}_{b_j}^w) - \mathbf{p}_c^b))$$



Monocular Visual-Inertial Odometry

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

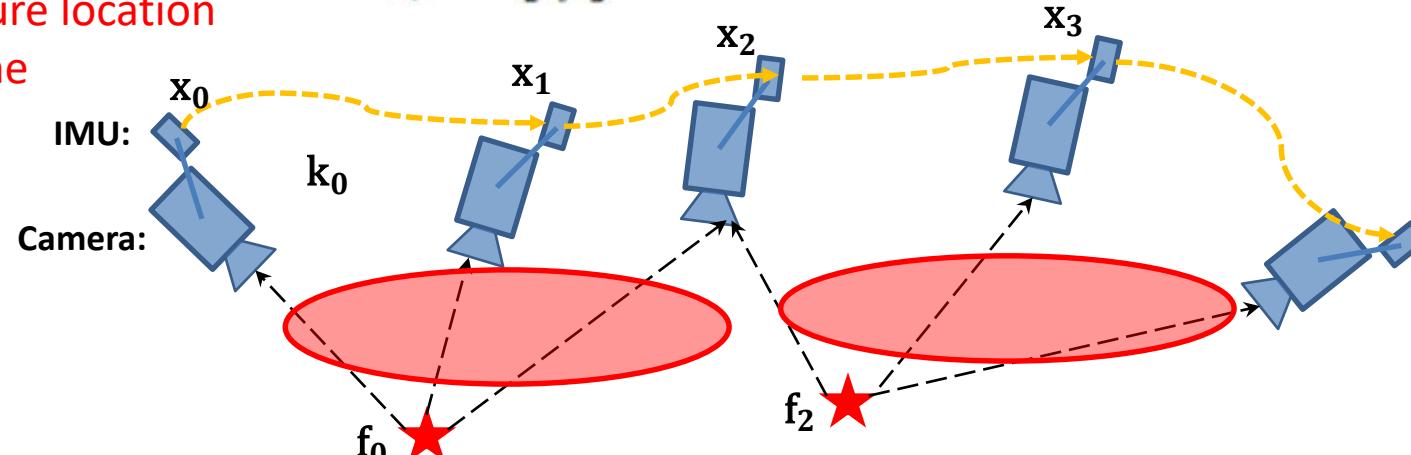
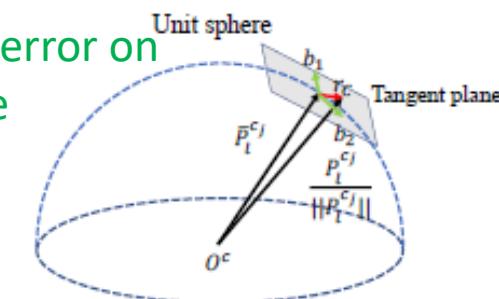
$$r_c(\hat{z}_l^{c_j}, \mathcal{X}) = [\mathbf{b}_1 \ \mathbf{b}_2]^T \cdot (\bar{\mathcal{P}}_l^{c_j} - \frac{\mathcal{P}_l^{c_j}}{\|\mathcal{P}_l^{c_j}\|})$$

$\bar{\mathcal{P}}_l^{c_j} = \pi_c^{-1} \left(\begin{bmatrix} \hat{u}_l^{c_j} \\ \hat{v}_l^{c_j} \end{bmatrix} \right)$ Unit vector of feature from 2D measurement

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

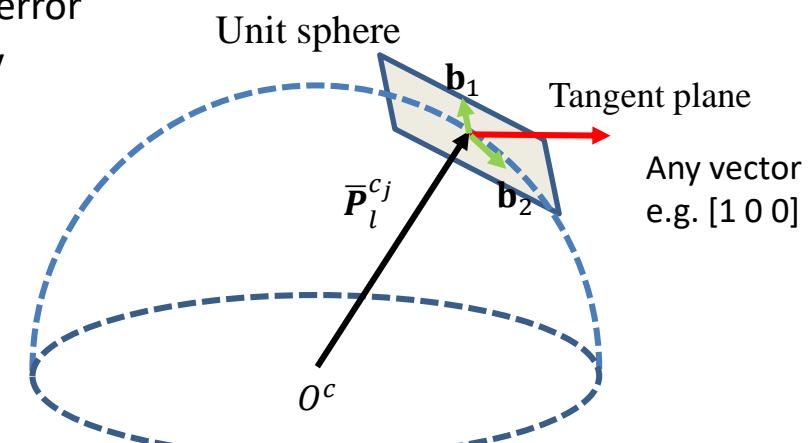
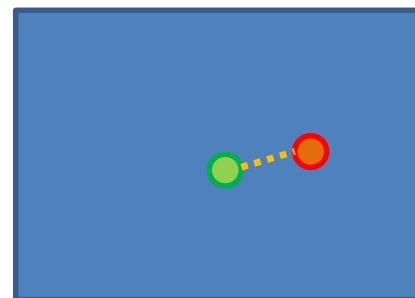
Predicted feature location
in camera frame

Reprojection error on
tangent plane



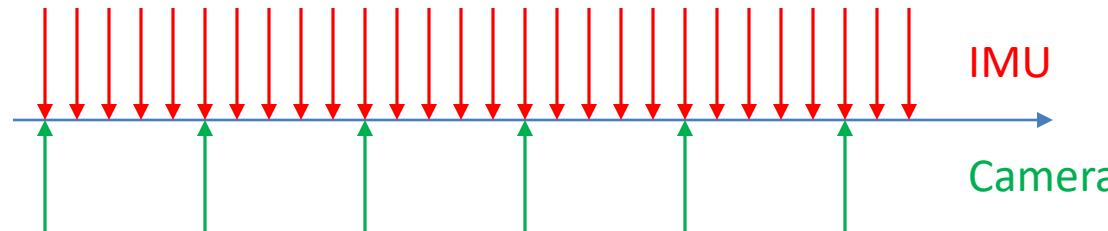
Monocular Visual-Inertial Odometry

- Vision measurement residual
 - Spherical camera model
 - Finding two basis vectors on the tangent plane
 - Choose any vector not parallel with $\bar{P}_l^{c_j}$, e.g. [1 0 0]
 - $\mathbf{b}_1 = \text{normalize}(\bar{P}_l^{c_j} \times [1 0 0])$
 - $\mathbf{b}_2 = \text{normalize}(\bar{P}_l^{c_j} \times \mathbf{b}_1)$
- Spherical vs. pinhole camera models
 - Different ways to define the reprojection error
 - Able to model cameras with arbitrary FOV

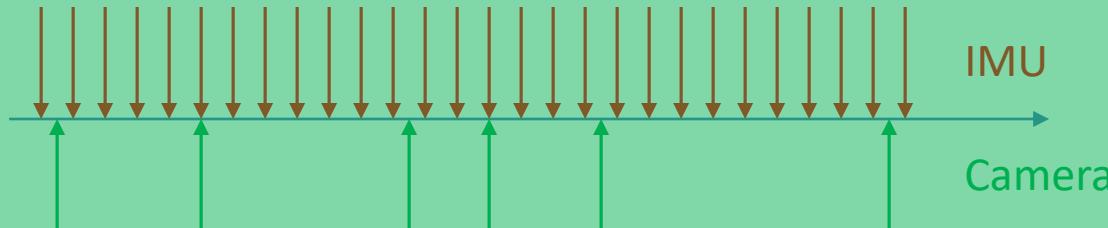


Review: Synchronization

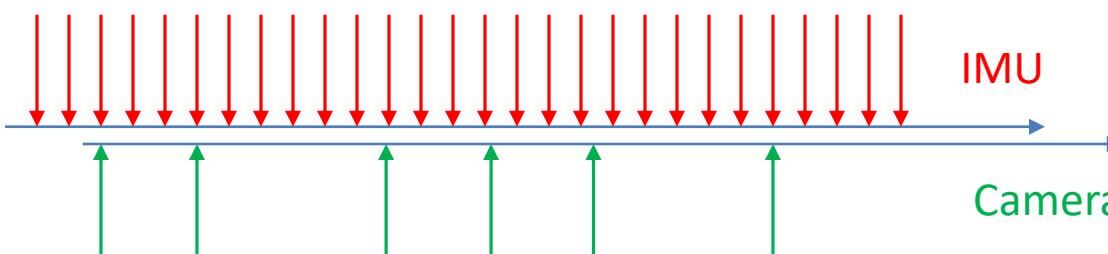
- Best: Sensors are hardware-triggered



- OK: Sensors have the same clock (e.g. running on the same system clock or have global clock correction) but capture data at different times

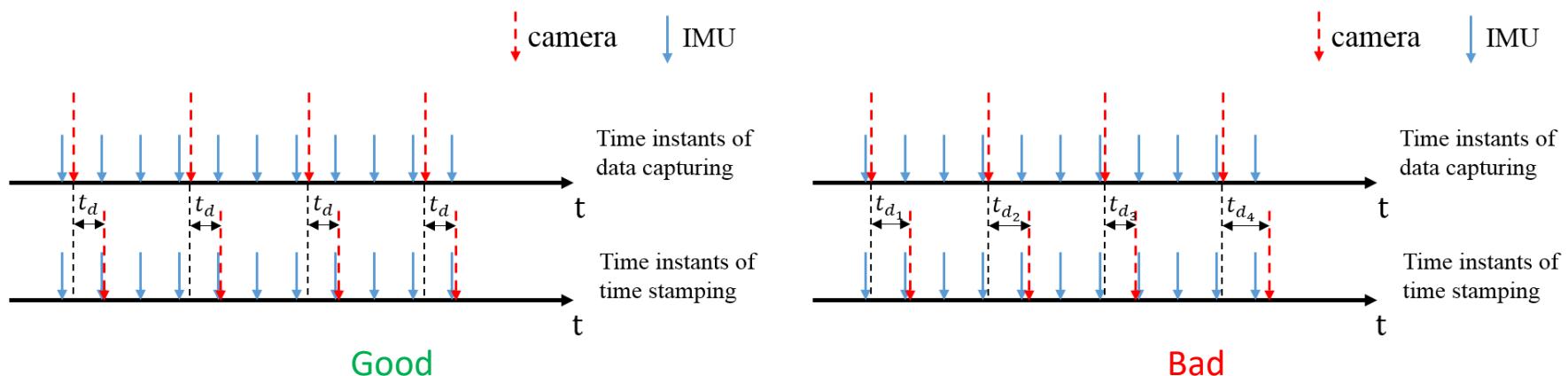


- Bad: Sensors have different clocks (e.g. each sensor has its own oscillator)



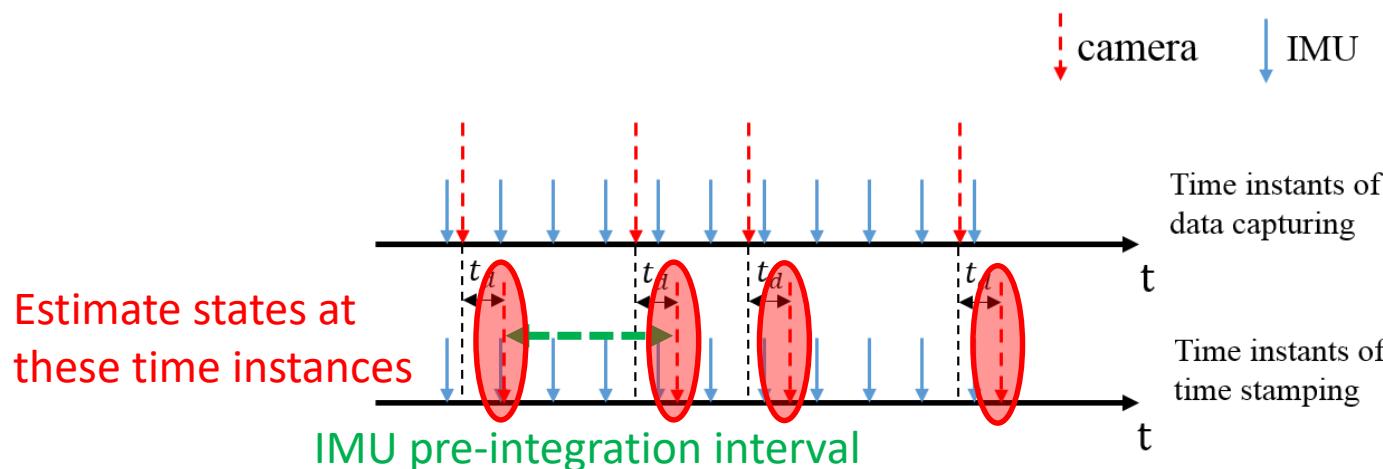
Review: Timestamps

- **Timestamp:** how the time for each sensor measurement is tagged
- **Best:** timestamping is done at data capture
- **OK:** fixed latency for time stamping
 - e.g. time is tagged on low-level hardware after some fixed-duration data processing, and will not be affected by any dynamic OS scheduling tasks
- **Bad:** variable latency in time stamping
 - e.g. plug two sensors into USB ports and time stamp according to the PC time. Time stamping is affected by data transmission latency from the sensor to PC



Monocular Visual-Inertial Odometry

- Temporal calibration
 - Calibrate the fixed latency t_d occurred during time stamping
 - Change the IMU pre-integration interval to the interval between two image timestamps
 - Linear incorporation of IMU measurements to obtain the IMU reading at image time stamping
 - Estimates states (position, orientation, etc.) **at image time stamping**



Monocular Visual-Inertial Odometry

- Vision measurement residual for temporal calibration

- Feature velocity on image plane

- feature l moves at speed V_l^k from image k to $k+1$ in short time period $[t_k, t_{k+1}]$

$$\mathbf{V}_l^k = (\begin{bmatrix} u_l^{k+1} \\ v_l^{k+1} \end{bmatrix} - \begin{bmatrix} u_l^k \\ v_l^k \end{bmatrix}) / (t_{k+1} - t_k)$$

- Visual measurement residual with time offset

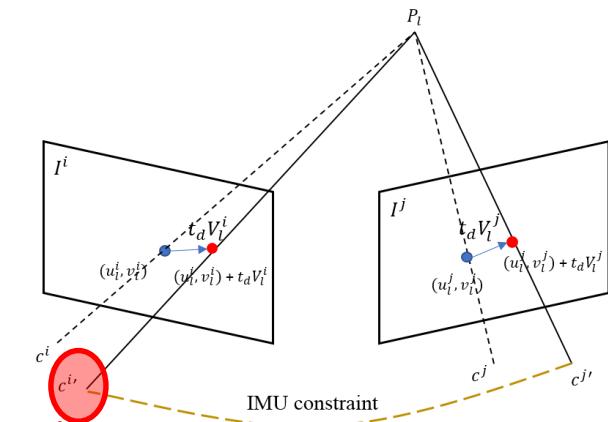
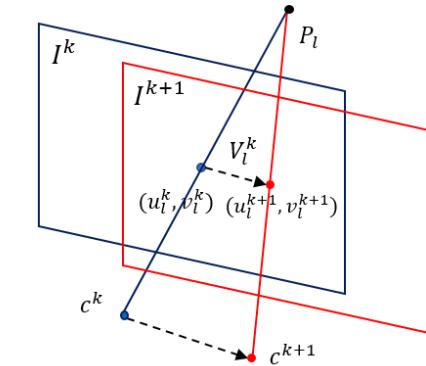
- New state variable t_d , and estimate states $(c^{i'}, c^{j'})$ at time stamping

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

$$\bar{\mathcal{P}}_l^{c_j} = \pi_c^{-1} \left(\begin{bmatrix} \hat{u}_l^{c_j} \\ \hat{v}_l^{c_j} \end{bmatrix} + (t_d \vec{V}_l^{c_j}) \right)$$

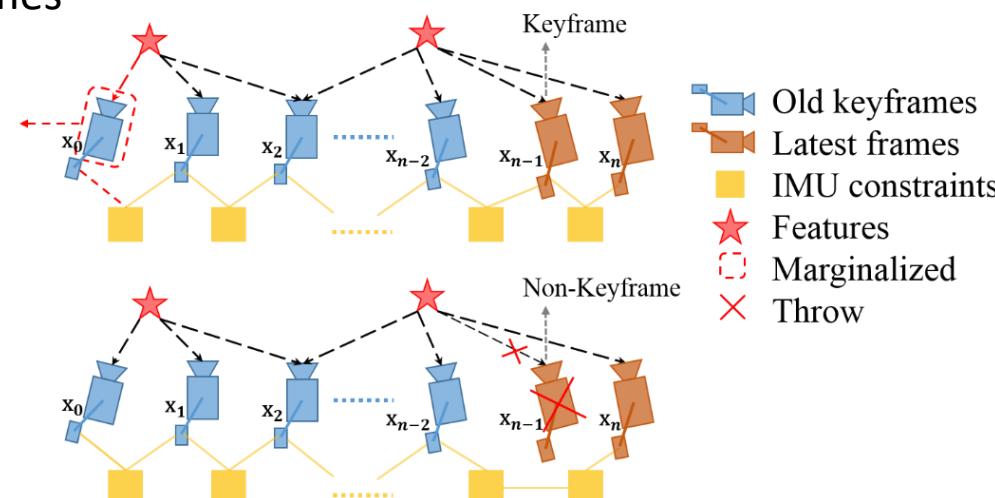
$$\begin{aligned} \mathcal{P}_l^{c_j} = \mathbf{R}_b^c (\mathbf{R}_w^{b_j} (\mathbf{R}_{b_i}^w (\mathbf{R}_c^b \frac{1}{\lambda_l} \pi_c^{-1} (\begin{bmatrix} u_l^{c_i} \\ v_l^{c_i} \end{bmatrix} + t_d \vec{V}_l^{c_i}) \\ + \mathbf{p}_c^b) + \mathbf{p}_{b_i}^w - \mathbf{p}_{b_j}^w) - \mathbf{p}_c^b)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

“Virtual image” at time stamping



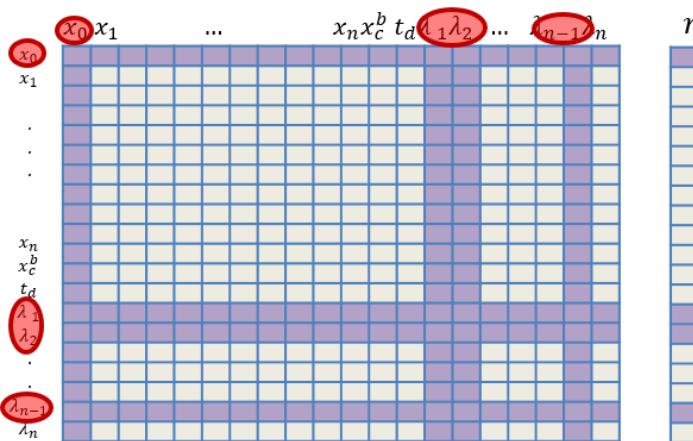
Monocular Visual-Inertial Odometry

- Marginalization
 - Bound computation complexity to a sliding window of states
 - Basic principles:
 - Add all frames into the sliding window, and remove non-keyframes after the nonlinear optimization
 - keep as many keyframes with sufficient parallax as possible
 - Maintain matrix sparsity by throwing away visual measurements from non-keyframes

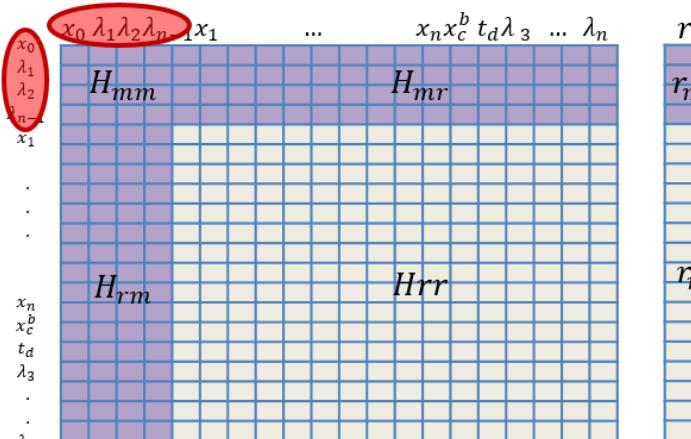
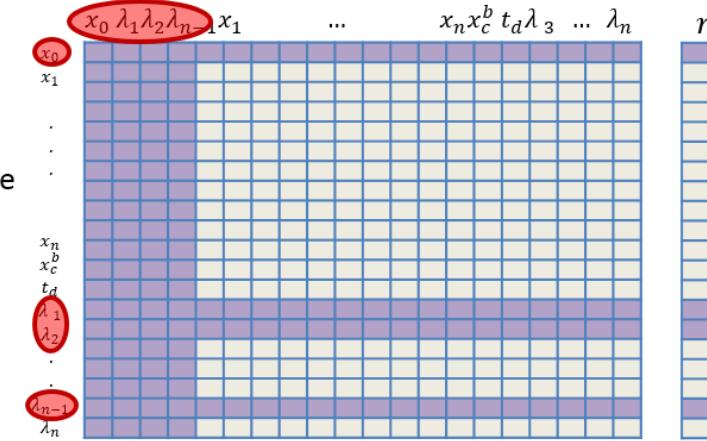


Monocular Visual-Inertial Odometry

- Marginalization via Schur complement on information matrix



Column rearrange



Row rearrange

$$\begin{aligned} \begin{bmatrix} H_{mm} & H_{mr} \\ H_{rm} & H_{rr} \end{bmatrix} \begin{bmatrix} x_m \\ x_r \end{bmatrix} &= \begin{bmatrix} r_m \\ r_r \end{bmatrix} \\ \underbrace{(H_{rr} - H_{rm}H_{mm}^{-1}H_{mr})}_{H_p} x_r &= \underbrace{r_r - H_{rm}H_{mm}^{-1}r_m}_{r_p} \end{aligned}$$

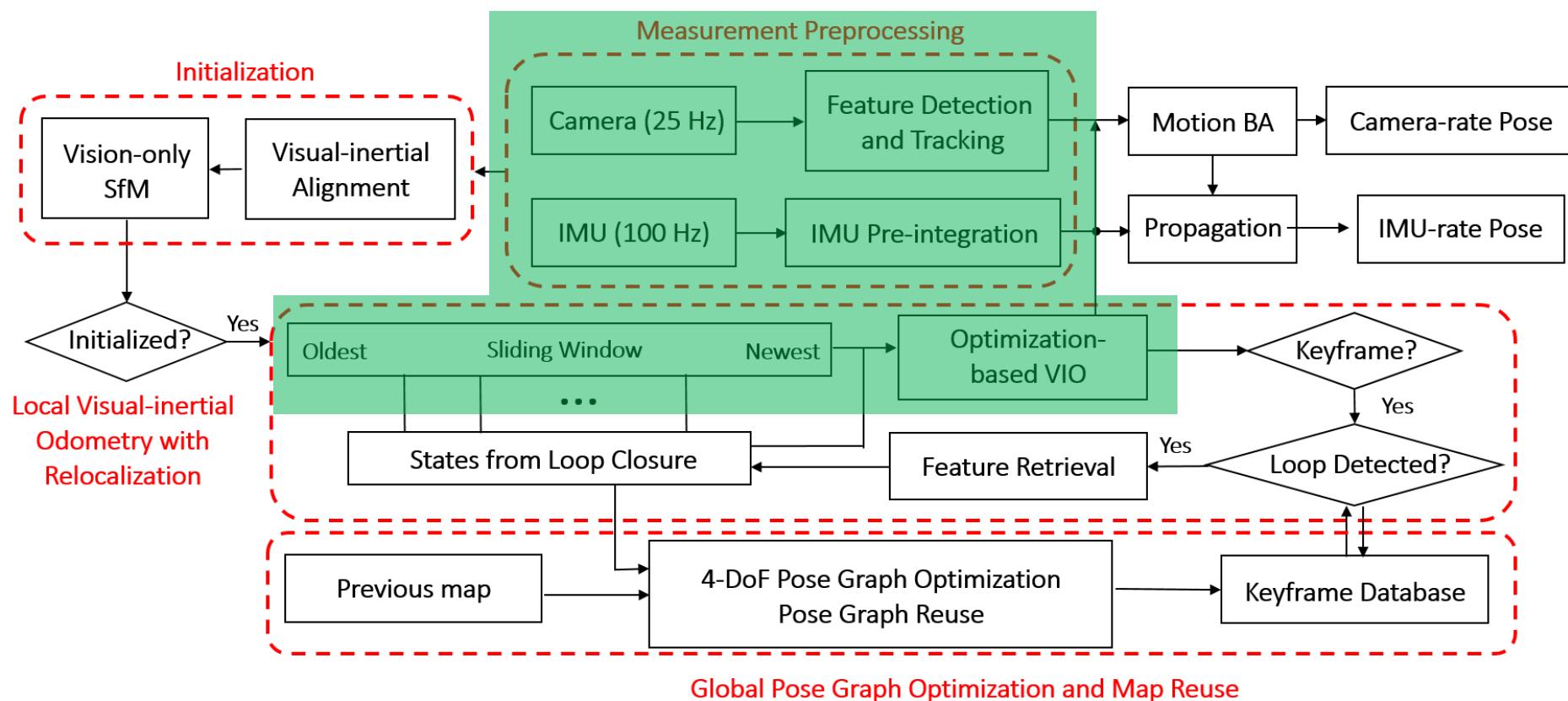
Monocular Visual-Inertial Odometry

- Solving the nonlinear system
 - Minimize residuals from all sensors

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

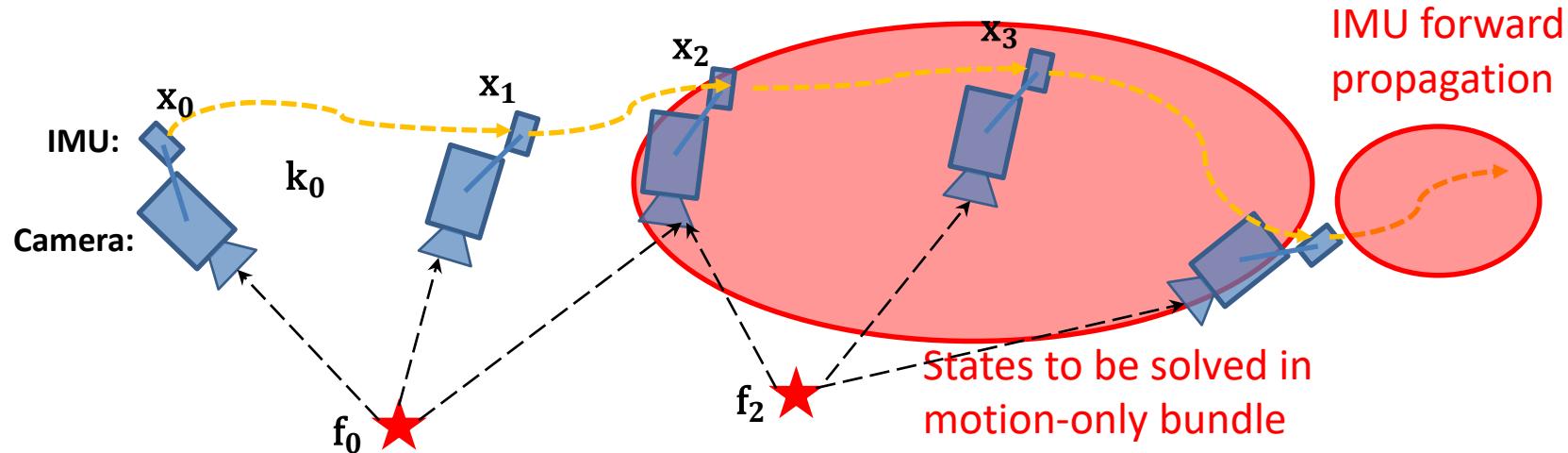
Monocular Visual-Inertial SLAM

- System diagram



Monocular Visual-Inertial Odometry

- Speeding up
 - The sliding window monocular visual-inertial bundle adjustment runs at 10Hz
 - Motion-only visual-inertial bundle adjustment to boost up the state estimation 30Hz
 - IMU forward propagation to boost to 100Hz

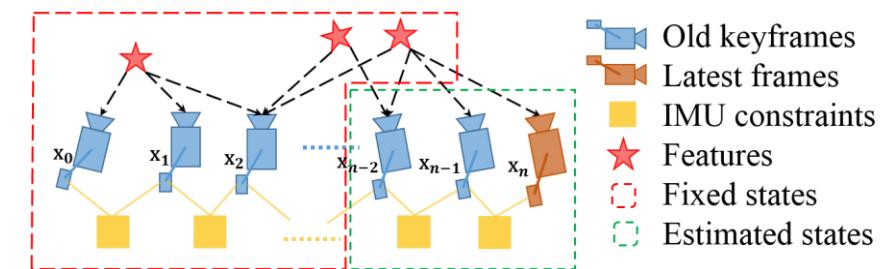


Monocular Visual-Inertial Odometry

- Motion-only visual-inertial bundle adjustment
 - Optimize **position, velocity, rotation** in a smaller windows, assuming all other quantities are fixed

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

- Prior in cost function is ignored



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

IMU measurement residual Vision measurement residual

Covariance from IMU pre-integration Pixel reprojection covariance

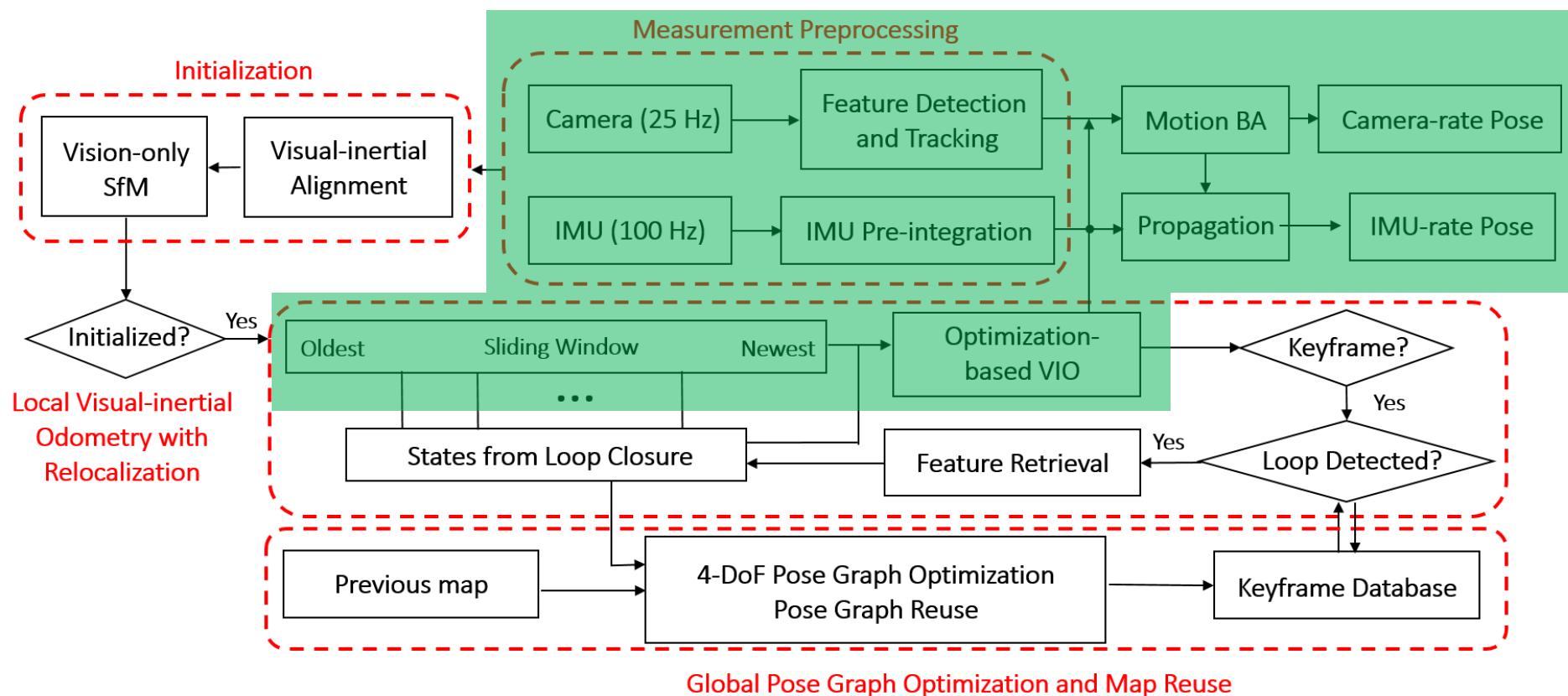
- Also solved using the Ceres Solver

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...

Monocular Visual-Inertial SLAM

- System diagram

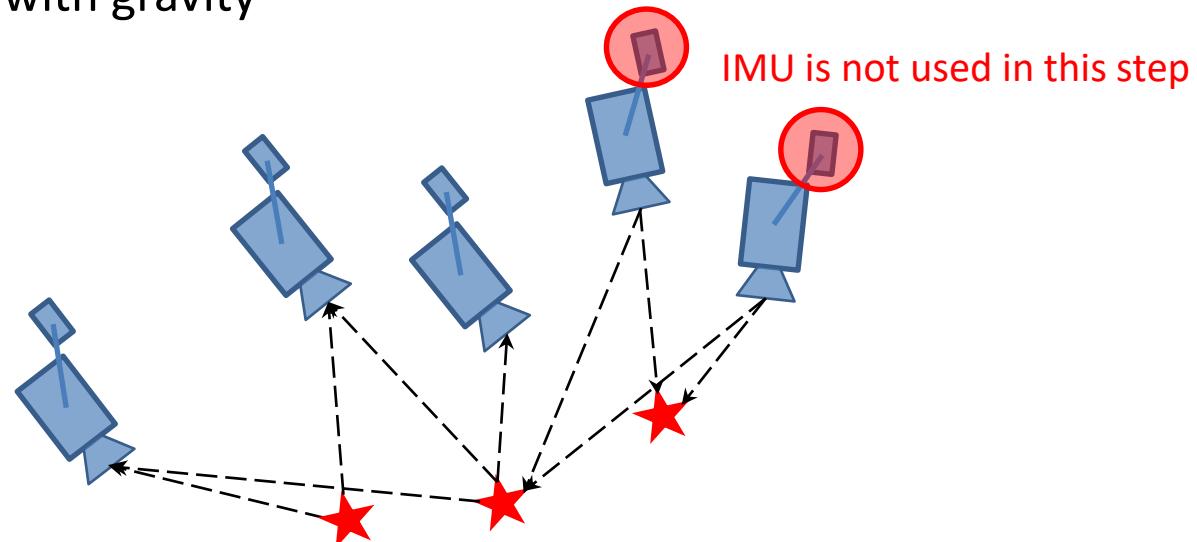


Estimator Initialization

- **Very, very, very** important for monocular visual-inertial systems
- Assumption 1: known camera-IMU extrinsic calibration during initialization
 - Does not need to be very accurate
 - Extrinsic calibration is refined in later nonlinear optimization
- Assumption 2: known accelerometer and gyroscope biases during initialization
 - Use zero values at power-up
 - Use prior values during failure recovery
 - Reasonable assumption due to slow varying nature of biases
- Pipeline
 - Monocular vision-only SFM in a local window
 - Visual-inertial alignment

Estimator Initialization

- Monocular vision-only structure-from-motion (SfM)
 - In a small window (10 frames, 1sec)
 - Up-to-scale, locally drift-free position estimates
 - Locally drift-free orientation estimates
 - Not aligned with gravity

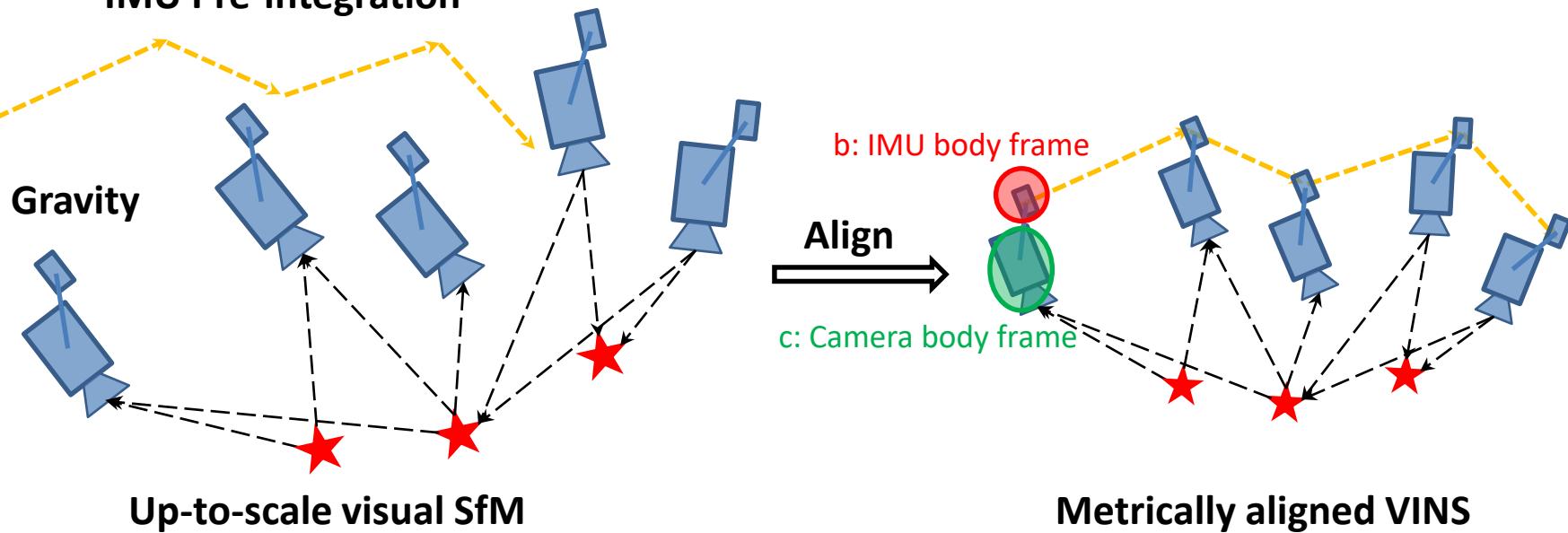


Estimator Initialization

- Visual-inertial alignment
 - Estimates **velocity** of each frame, **gravity** vector, and **scale**
 - Note the coordinate frames

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

IMU Pre-integration



Estimator Initialization

- Visual-inertial alignment
 - Linear measurement model

IMU Pre-integration

$$\hat{\mathbf{z}}_{b_{k+1}}^{b_k} = \begin{bmatrix} \hat{\alpha}_{b_{k+1}}^{b_k} \\ \mathbf{R}_{c_{k+1}}^{c_0} \mathbf{P}_b^c + \mathbf{R}_{c_k}^{c_0} \mathbf{P}_b^c \\ \beta_{b_{k+1}}^{b_k} \end{bmatrix}$$

Known values from vSfM
and extrinsic calibration

$$= \mathbf{H}_{b_{k+1}}^{b_k} \boldsymbol{\chi}_I + \mathbf{n}_{b_{k+1}}^{b_k}$$

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

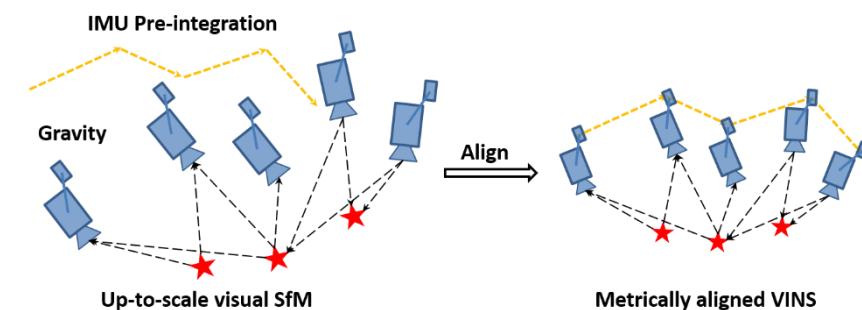
Up-to-scale translation

States to be initialized

- Solve a linear system from vSfM
 - Scale and rotate the vSfM

$$\boldsymbol{\chi}_I = [\mathbf{v}_{b_0}^{c_0}, \mathbf{v}_{b_1}^{c_0}, \dots, \mathbf{v}_{b_n}^{c_0}, \mathbf{g}^{c_0}, s]$$

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

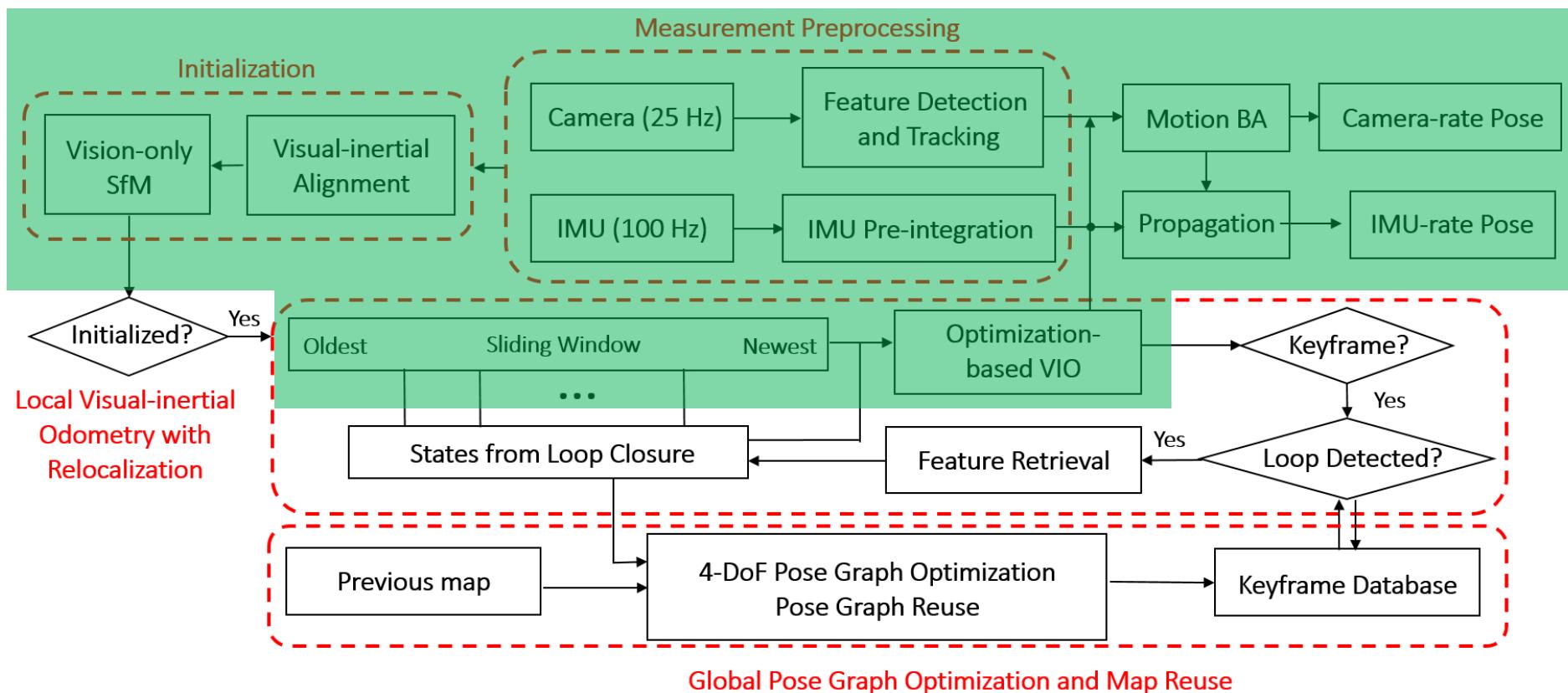


Estimator Initialization

- 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

Monocular Visual-Inertial SLAM

- System diagram



Visual-Inertial SLAM for Autonomous Drone

Monocular Visual-Inertial System (VINS-Mono)
on MAV Platform for Autonomous Flight

Tong Qin, Peiliang Li, Zhenfei Yang and Shaojie Shen



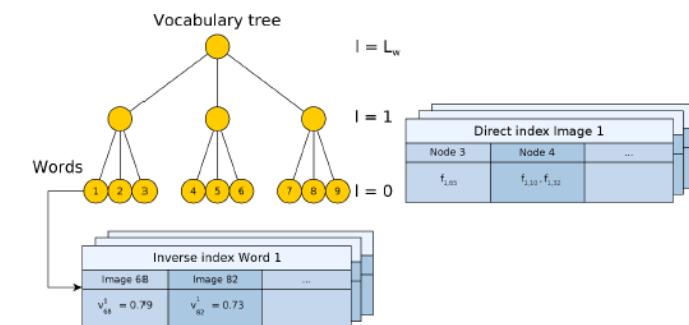
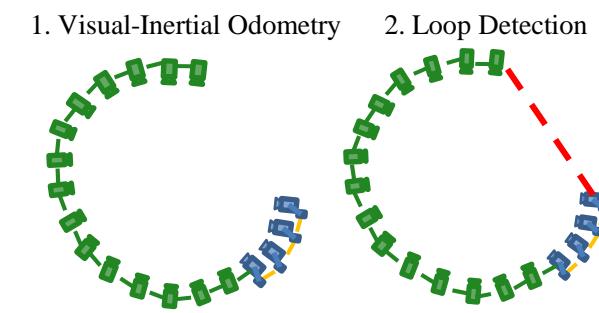
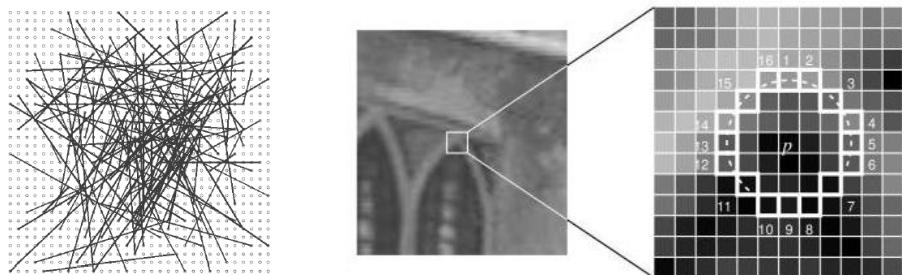
香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

HKUST
Aerial Robotics Group

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

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



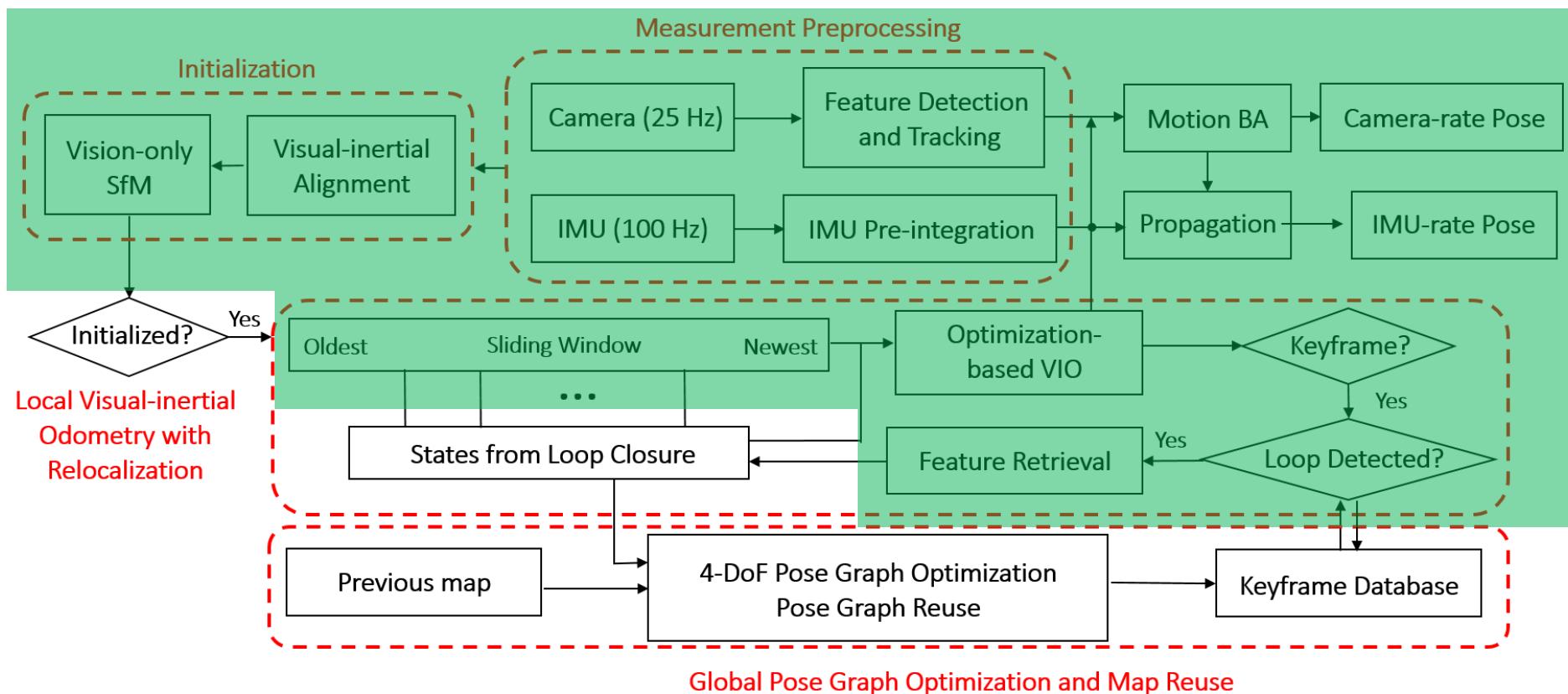
(a)



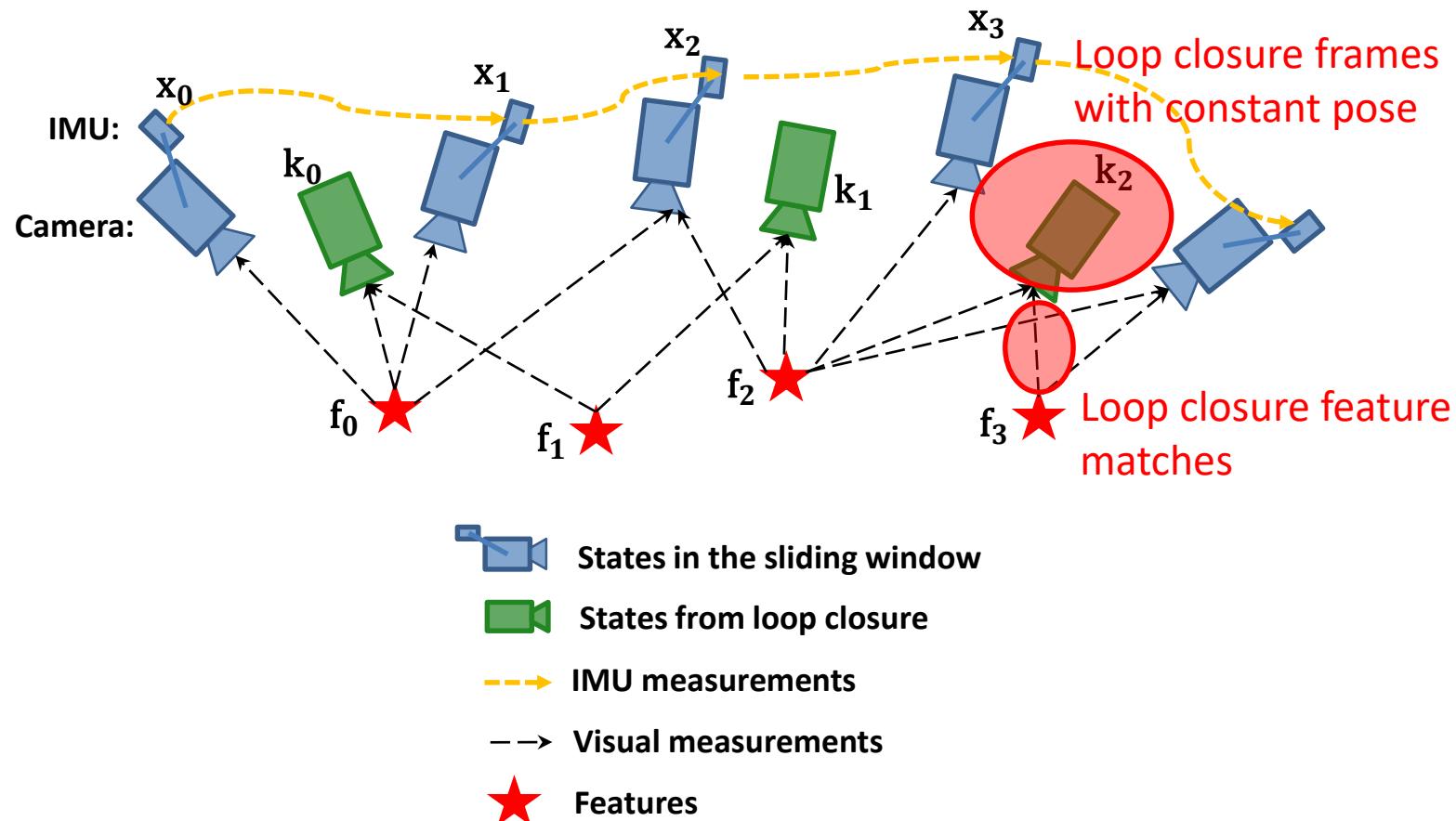
(b)

Monocular Visual-Inertial SLAM

- System diagram



Monocular Visual-Inertial Odometry with Relocalization

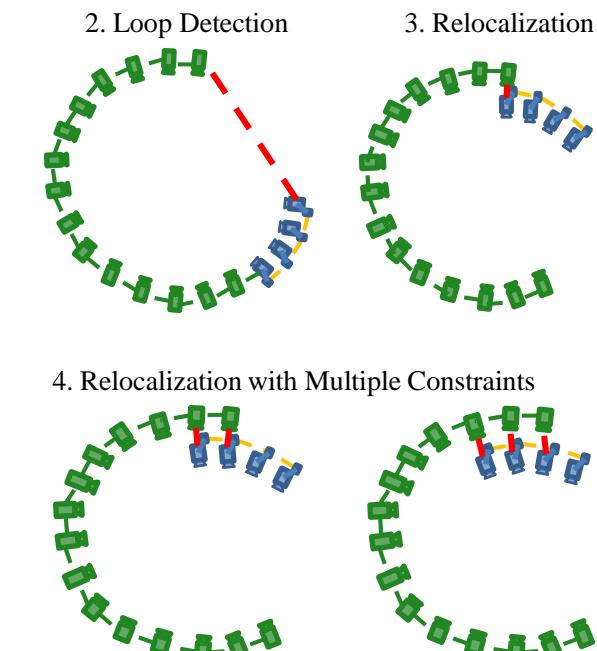


Monocular Visual-Inertial Odometry with Relocalization

- 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

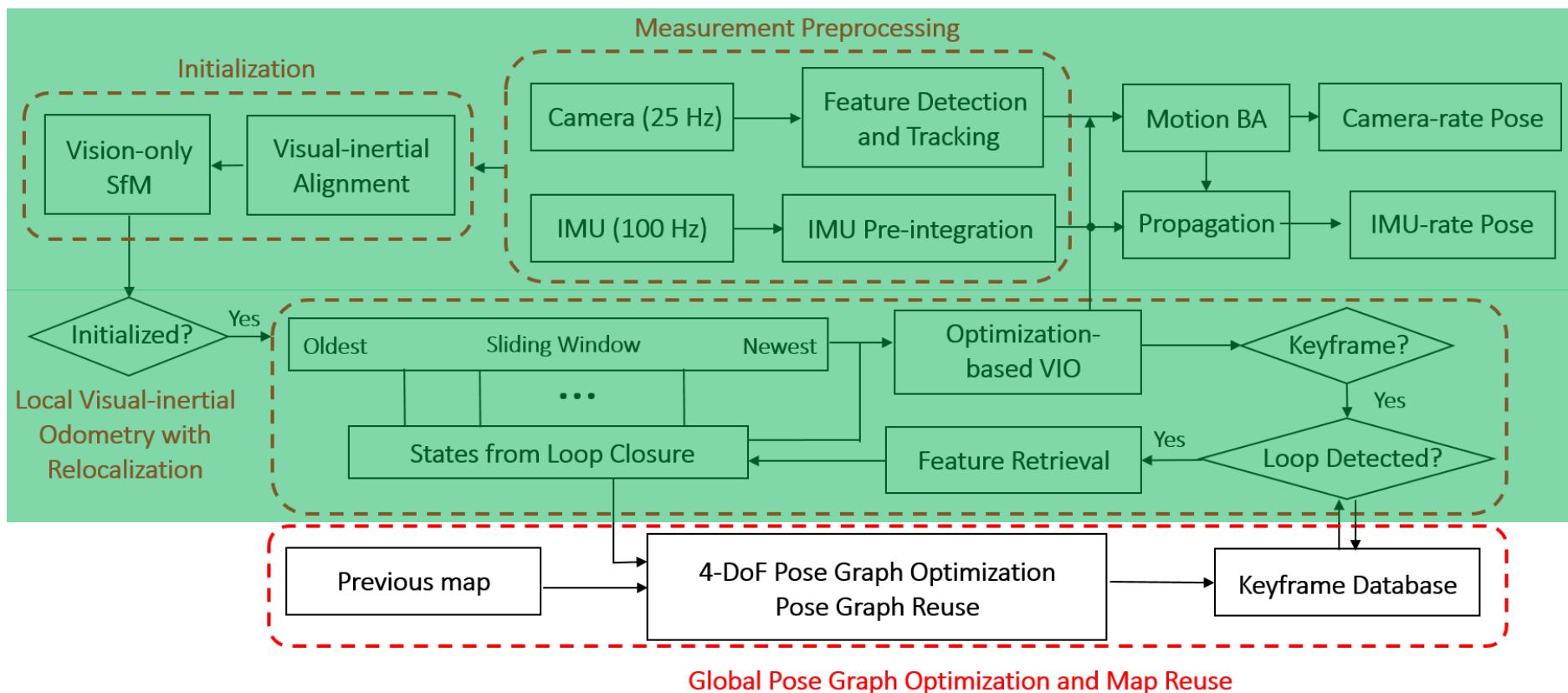
$$\begin{aligned}
 \text{VIO residuals} \quad \min_{\mathcal{X}} \left\{ \|\mathbf{r}_p - \mathbf{H}_p \mathcal{X}\|^2 + \sum_{k \in \mathcal{B}} \left\| \mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}_{b_{k+1}}^{b_k}, \mathcal{X}) \right\|_{\mathbf{P}_{b_{k+1}}^{b_k}}^2 + \right. \\
 \left. \sum_{(l,j) \in \mathcal{C}} \left\| \mathbf{r}_c(\hat{\mathbf{z}}_l^{c_j}, \mathcal{X}) \right\|_{\mathbf{P}_l^{c_j}}^2 + \sum_{(l,v) \in \mathcal{L}} \left\| \mathbf{r}_c(\hat{\mathbf{z}}_l^v, \mathcal{X}, \hat{\mathbf{q}}_v^w, \hat{\mathbf{p}}_v^w) \right\|_{\mathbf{P}_l^v}^2 \right\}, \quad (22)
 \end{aligned}$$

Loop closure vision measurement residual



Monocular Visual-Inertial SLAM

- System diagram



Global Pose Graph SLAM

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

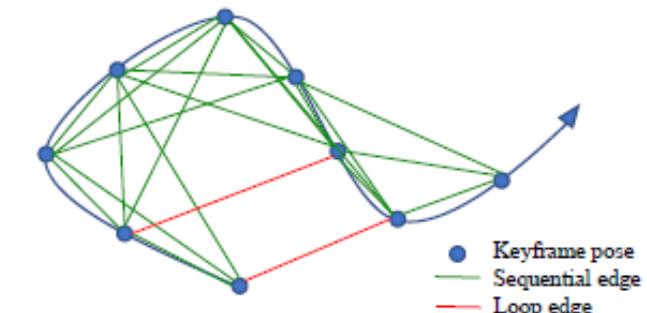
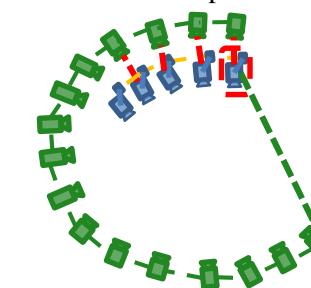
- Adding keyframes into pose graph

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

$$\hat{\psi}_{ij} = \hat{\psi}_j - \hat{\psi}_i$$

- 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:

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

Observable attitude from VIO

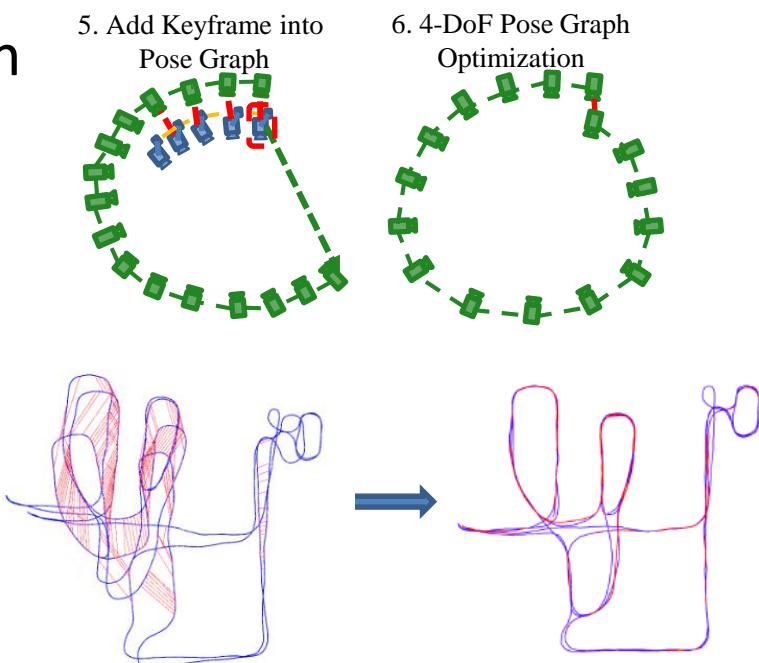
- Minimize the following cost function

- Sequential edge from VIO
- Loop closure edges
 - Huber norm for rejection of wrong loops

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

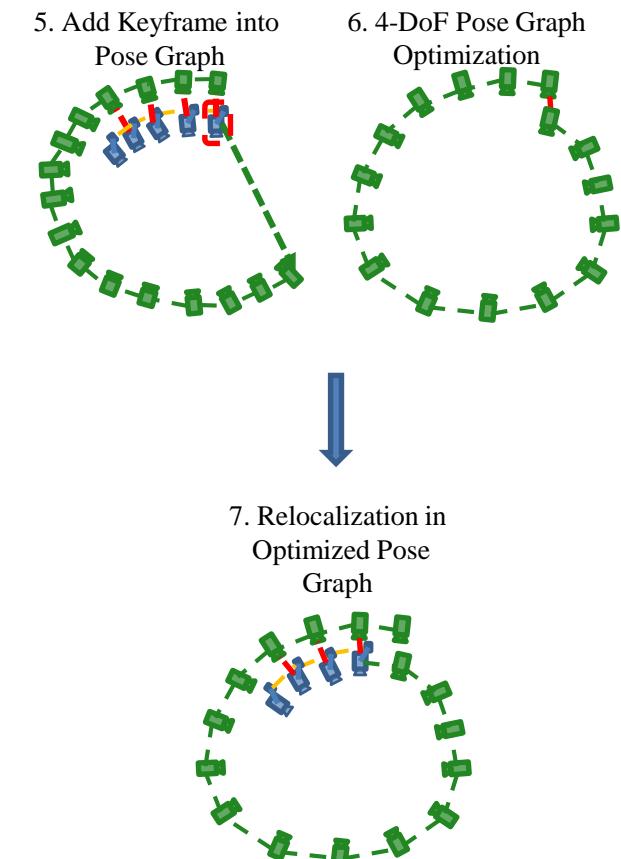
Sequential edges

Loop closure edges



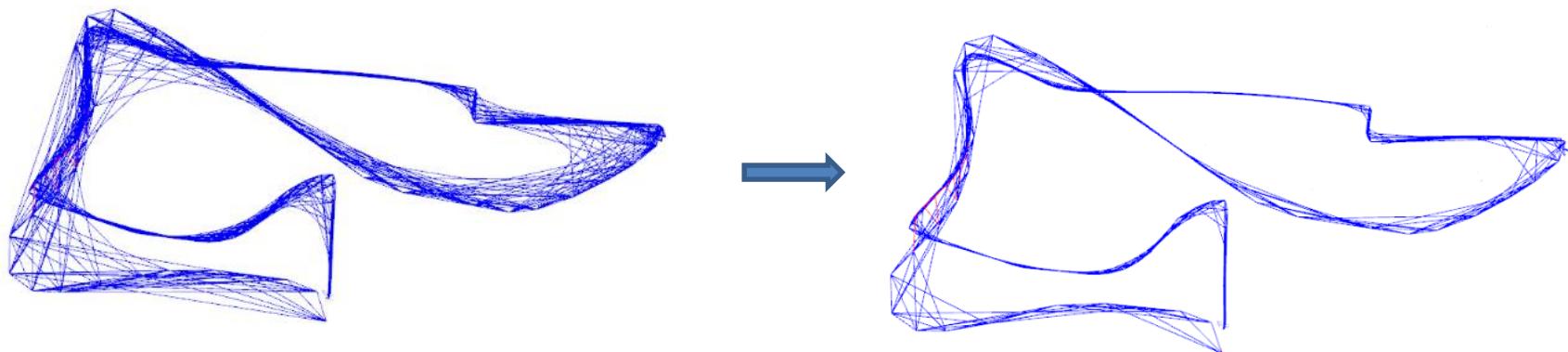
Global Pose Graph SLAM

- 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



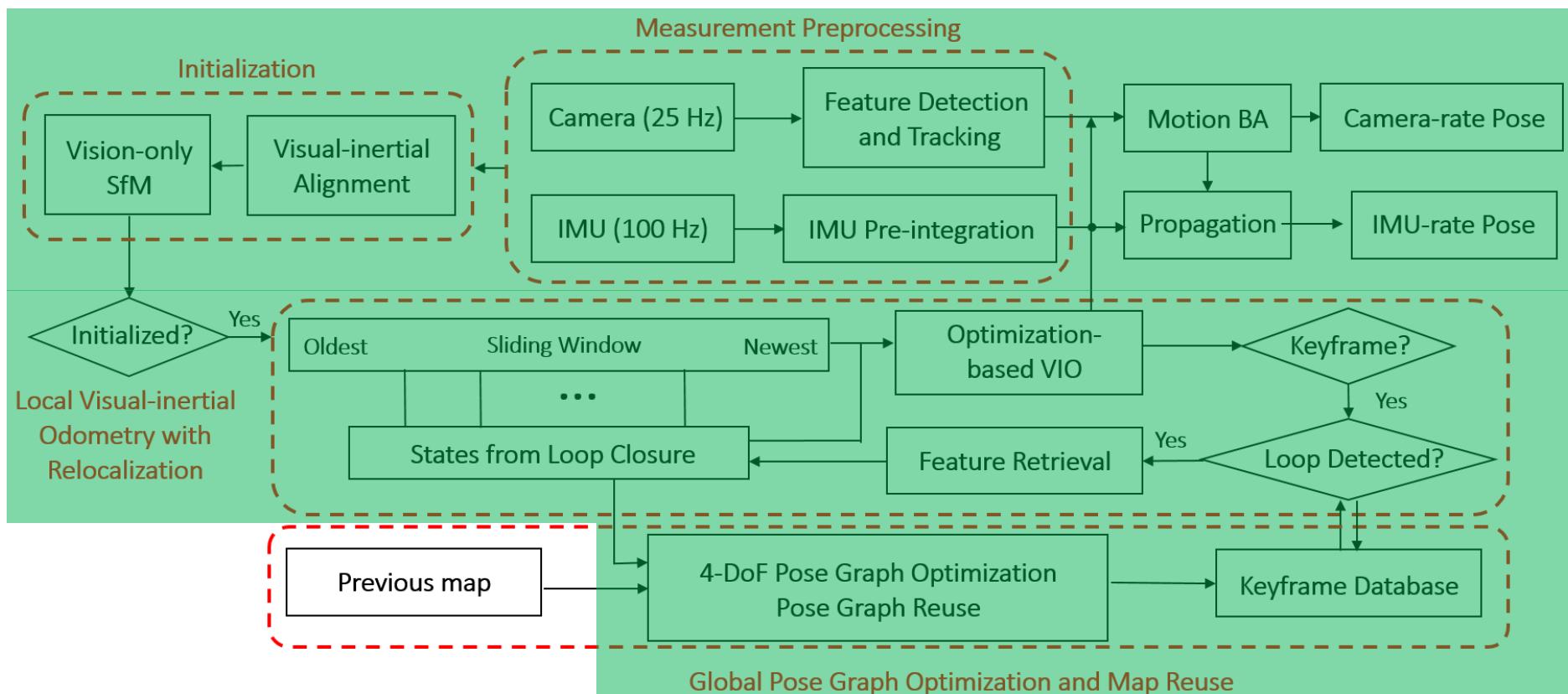
Global Pose Graph SLAM

- 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



Monocular Visual-Inertial SLAM

- System diagram



Visual-Inertial SLAM in Large-Scale Environment

II. Go out laboratory

Single camera: mvBluefox

IMU: DJI A3

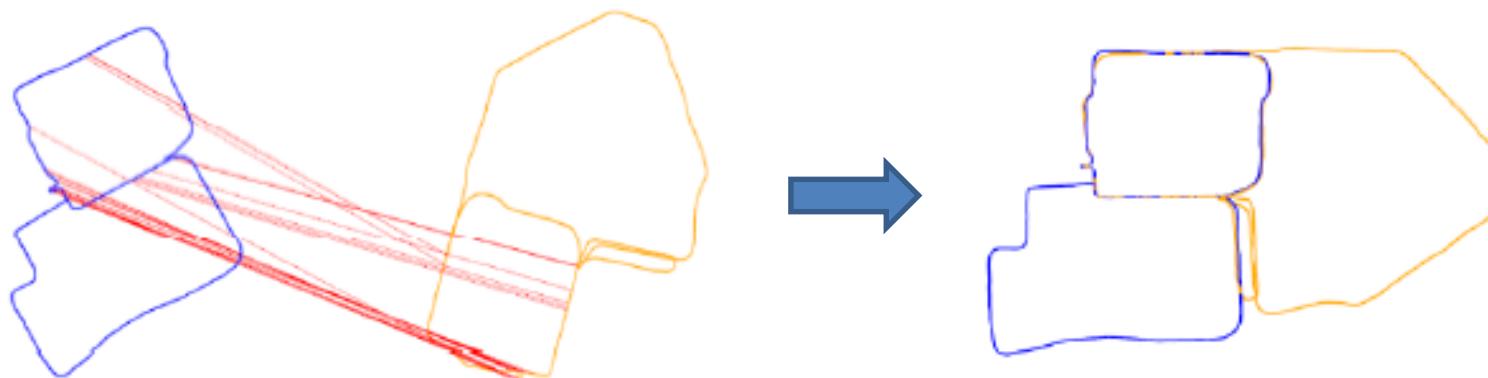


Pose Graph Reuse

- Pose graph saving
 - Every Keyframe
 - Index i , position $\hat{\mathbf{p}}_i^w$, orientation $\hat{\mathbf{q}}_i^w$, features' 2D location and descriptor $D(u, v, des)$
 - If i loops with v , we also save loop index v , relative translation $\hat{\mathbf{p}}_{iv}^i$, relative yaw angle $\hat{\phi}_{iv}$
 $[i, \hat{\mathbf{p}}_i^w, \hat{\mathbf{q}}_i^w, v, \hat{\mathbf{p}}_{iv}^i, \hat{\phi}_{iv}, \mathbf{D}(u, v, des)]$
- Pose graph loading
 - Build sequential edges
 - Connected with 4 previous keyframes
 - Build loop closure edges
 - According to loop index v , relative translation $\hat{\mathbf{p}}_{iv}^i$ and yaw angle $\hat{\phi}_{iv}$

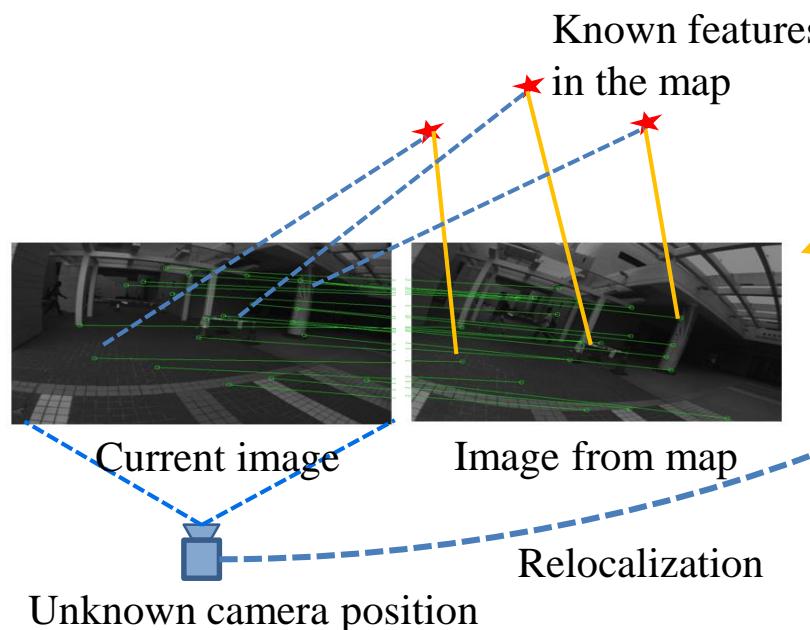
Pose Graph Reuse

- Pose graph merging
 - Load a previous-built map
 - Build a new map
 - Detect loop connections between two maps
 - Merge two map by pose graph optimization



Pose Graph Reuse

- Relocalization
 - Load previous-built map (aligned with Google Map)
 - The camera starts at an unknown position
 - Detect similar image view in the map
 - Once loop detected, relocate camera pose



Previously built map

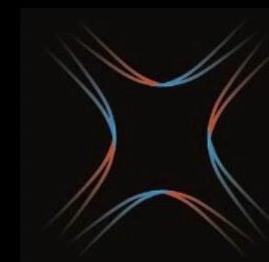
Pose Graph Reuse

Relocalization, Global Optimization and Map Merging
for Monocular Visual-Inertial SLAM

Tong Qin, Peiliang Li, and Shaojie Shen



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



香港科技大學—
大疆創新科技聯合實驗室
HKUST-DJI JOINT
INNOVATION LABORATORY

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

Remarks on Monocular Visual-Inertial SLAM

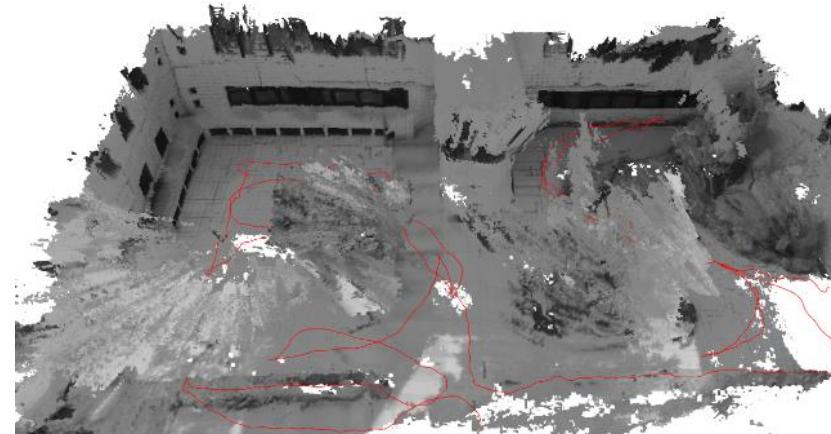
- Important factors
 - Access to raw camera data (especially for rolling shutter cameras)
 - Sensor synchronization and timestamps
 - Camera-IMU rotation
 - Estimator initialization
- Not-so-important factors
 - Camera-IMU translation
 - Types of features (we use the simplest corner+KLT)
 - Quality of feature tracking (outlier is acceptable)
- Failures – need more engineering treatment
 - Long range scenes (aerial vehicles)
 - Constant velocity (ground vehicle)
 - Pure rotation (augmented reality)
- Be aware of computational power requirement

Remarks on Monocular Visual-Inertial SLAM

- 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
 - Intrinsic and extrinsic calibration of IMU, rolling shutter, etc.
 - Synchronization issues
 - Poor sensors and manufacturing variations
 - Insufficient computing power
 - Big players are doing a good job

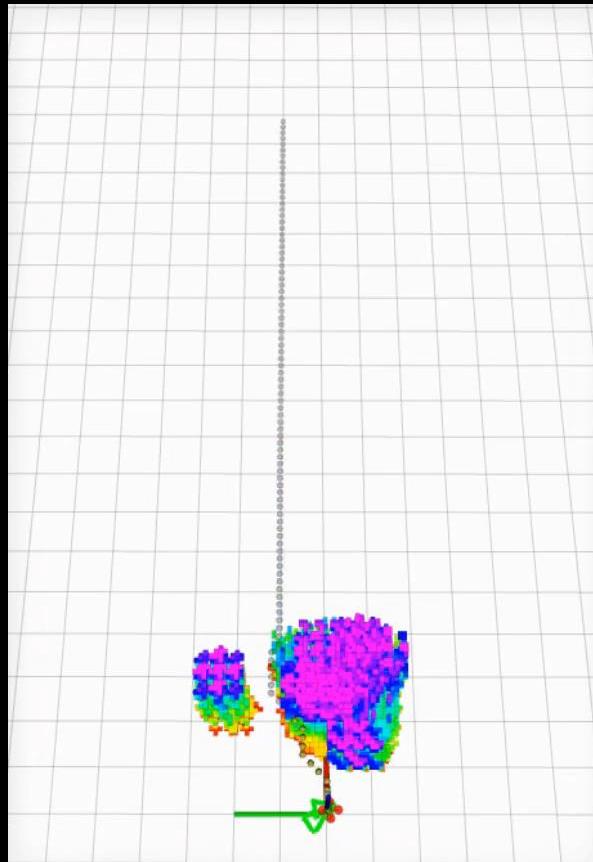
Remarks on Monocular Visual-Inertial SLAM

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



Dense Mapping, Trajectory Planning, and Navigation

Indoor Experiment 2:



2x

Trajectory length: 18.6m
Total number of replans: 125
Average computing time: 43ms
Average snap: 1.21m/s⁴

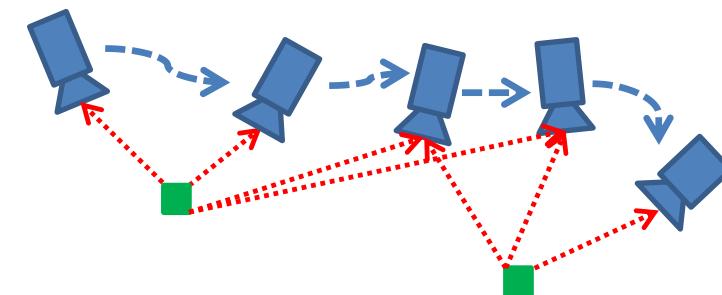
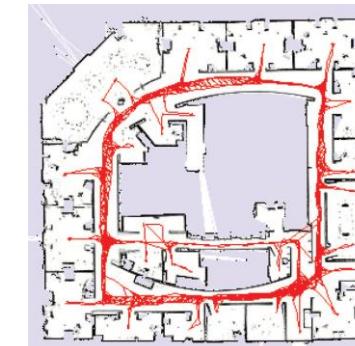


2x

Summary

$$P(x \mid y, z) = \frac{P(y \mid x, z) P(x \mid z)}{P(y \mid z)}$$

- The Basics
- 2D Graph-Based SLAM
- Monocular Visual-Inertial SLAM



Logistics

- Project 3 Phase 1 due this Friday (28 April).
- Project 3 Phase 2 due this Friday (28 April).
- Project 3 Phase 3 is released, not the hard time comes...
- Give a try to VINS-Mono after completing all your projects.
- This is the last lecture.
- Please help with the course evaluation.

Thanks for taking ELEC5660

and enjoy your projects ☺

Appendix: Quaternion

Definitions of Quaternion

- Quaternion

$$Q \stackrel{\text{def}}{=} q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \stackrel{\text{def}}{=} q_0 + \mathbf{q} \stackrel{\text{def}}{=} \begin{bmatrix} q_0 \\ \mathbf{q} \end{bmatrix}$$

where $\mathbf{i} \circ \mathbf{i} = \mathbf{j} \circ \mathbf{j} = \mathbf{k} \circ \mathbf{k} \stackrel{\text{def}}{=} -1; \mathbf{i} \circ \mathbf{j} \stackrel{\text{def}}{=} \mathbf{k}; \mathbf{j} \circ \mathbf{k} \stackrel{\text{def}}{=} \mathbf{i}; \mathbf{k} \circ \mathbf{i} \stackrel{\text{def}}{=} \mathbf{j}$

- Conjugate

$$\bar{Q} \stackrel{\text{def}}{=} q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k} \stackrel{\text{def}}{=} q_0 - \mathbf{q} \stackrel{\text{def}}{=} \begin{bmatrix} q_0 \\ -\mathbf{q} \end{bmatrix}$$

- Norm $\|Q\| \stackrel{\text{def}}{=} Q \circ \bar{Q}$

- Inverse $Q^{-1} \stackrel{\text{def}}{=} \frac{\bar{Q}}{\|Q\|}$

Properties of Quaternion

- Property 1

$$\mathbf{A} = a_0 + \mathbf{a}, \quad \mathbf{B} = b_0 + \mathbf{b} \quad \text{particularly}$$

$$\mathbf{A} \circ \mathbf{B} = (a_0 b_0 - \mathbf{a} \cdot \mathbf{b}) + (a_0 \mathbf{b} + \mathbf{a} b_0 + \mathbf{a} \times \mathbf{b})$$

$$(0 + \mathbf{a}) \circ (0 + \mathbf{b}) = -\mathbf{a} \cdot \mathbf{b} + \mathbf{a} \times \mathbf{b}$$

- Property 2 $(\mathbf{A} \circ \mathbf{B}) \circ \mathbf{C} = \mathbf{A} \circ (\mathbf{B} \circ \mathbf{C})$

- Property 3 $\overline{\mathbf{A} \circ \mathbf{B}} = \overline{\mathbf{B}} \circ \overline{\mathbf{A}}$

- Property 4 $\mathbf{A} = a_0 + a_1 \mathbf{i} + a_2 \mathbf{j} + a_3 \mathbf{k}$
 $\mathbf{B} = b_0 + b_1 \mathbf{i} + b_2 \mathbf{j} + b_3 \mathbf{k}$

Exactly what
we need for
representing
rotations!!

$$\mathbf{A} \circ \mathbf{B} = \underbrace{\begin{bmatrix} a_0 & -a_1 & -a_2 & -a_3 \\ a_1 & a_0 & -a_3 & a_2 \\ a_2 & a_3 & a_0 & -a_1 \\ a_3 & -a_2 & a_1 & a_0 \end{bmatrix}}_{\mathcal{L}(\mathbf{A})} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \underbrace{\begin{bmatrix} b_0 & -b_1 & -b_2 & -b_3 \\ b_1 & b_0 & b_3 & -b_2 \\ b_2 & -b_3 & b_0 & b_1 \\ b_3 & b_2 & -b_1 & b_0 \end{bmatrix}}_{\mathcal{R}(\mathbf{B})} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Properties of Quaternion

- Product of Quaternions $\mathbf{A} \circ \mathbf{B} = \mathcal{L}(\mathbf{A}) \begin{bmatrix} b_0 \\ \mathbf{b} \end{bmatrix} = \mathcal{R}(\mathbf{B}) \begin{bmatrix} a_0 \\ \mathbf{a} \end{bmatrix}$

$$\mathcal{L}(\mathbf{A}) = \begin{bmatrix} a_0 & -a_1 & -a_2 & -a_3 \\ a_1 & a_0 & -a_3 & a_2 \\ a_2 & a_3 & a_0 & -a_1 \\ a_3 & -a_2 & a_1 & a_0 \end{bmatrix} = \begin{bmatrix} a_0 & -\mathbf{a}^T \\ \mathbf{a} & a_0 \mathbf{I}_3 + \hat{\mathbf{a}} \end{bmatrix}$$

$$\mathcal{R}(\mathbf{B}) = \begin{bmatrix} b_0 & -b_1 & -b_2 & -b_3 \\ b_1 & b_0 & b_3 & -b_2 \\ b_2 & -b_3 & b_0 & b_1 \\ b_3 & b_2 & -b_1 & b_0 \end{bmatrix} = \begin{bmatrix} b_0 & -\mathbf{b}^T \\ \mathbf{b} & b_0 \mathbf{I}_3 - \hat{\mathbf{a}} \end{bmatrix}$$

Unit Quaternion and Rotation

Recall

$$\begin{bmatrix} 0 \\ \mathbf{p}' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos \frac{\theta}{2} \\ \sin \frac{\theta}{2} \mathbf{u} \end{bmatrix}}_{\mathbf{P}'} \circ \underbrace{\begin{bmatrix} 0 \\ \mathbf{p} \end{bmatrix}}_{\mathbf{P}} \circ \underbrace{\begin{bmatrix} \cos \frac{\theta}{2} \\ -\sin \frac{\theta}{2} \mathbf{u} \end{bmatrix}}_{\overline{\mathbf{Q}}}$$

$$\mathbf{P}' \stackrel{\text{def}}{=} 0 + \mathbf{p}'; \mathbf{P} \stackrel{\text{def}}{=} 0 + \mathbf{p}; \quad \boxed{\mathbf{Q} \stackrel{\text{def}}{=} \cos \frac{\theta}{2} + \sin \frac{\theta}{2} \mathbf{u}}$$

If using quaternion Q to represent a rotation:

Basic principle: $\mathbf{P}' = \mathbf{Q} \circ \mathbf{P} \circ \overline{\mathbf{Q}}$

- \mathbf{P}', \mathbf{P} are respectively the coordinates of the post-rotation vector and prior-rotation vector, both in the **same** frame.
- \mathbf{Q} is the rotation quaternion associated with the rotation

Unit Quaternion and Rotation

Basic principle: $P' = Q \circ P \circ \bar{Q}$

Property 1 Constant vector: $P^b = \bar{Q} \circ P \circ R$

- P^b, P are respectively the coordinates in the body frame and the world frame, of the **same** vector
- Can be interpreted as that the vector is rotating in the opposite direction, then call the first result

Unit Quaternion and Rotation

Basic principle: $P' = Q \circ P \circ \bar{Q}$

Property 2 Two sequent rotations

- Case 1

Q_1, Q_2 are the two rotation quaternions where the rotation axes are both represented in the **initial** frame

$$\left. \begin{aligned} P' &= Q_1 \circ P \circ \bar{Q}_1 \\ P'' &= Q_2 \circ P' \circ \bar{Q}_2 = \underbrace{Q_2 \circ Q_1}_{Q} \circ P \circ \bar{Q}_1 \circ \bar{Q}_2 \end{aligned} \right\} \Rightarrow Q = Q_2 \circ Q_1$$

- Case 2

Q_1, Q_2 are the two rotation quaternions where the rotation axis of Q_2 is represented in the frame obtained by performing Q_1 .

$$P' = Q_1 \circ P \circ \bar{Q}_1$$

$$\begin{aligned} P'' &= (\underline{Q_1 \circ Q_2 \circ \bar{Q}_1}) \circ P' \circ \overline{Q_1 \circ Q_2 \circ \bar{Q}_1} = (Q_1 \circ Q_2 \circ \bar{Q}_1) \circ Q_1 \circ P \circ \bar{Q}_1 \circ \overline{Q_1 \circ Q_2 \circ \bar{Q}_1} = \underbrace{Q_1 \circ Q_2 \circ P \circ \overline{Q_1 \circ Q_2}}_Q \\ &\Rightarrow Q = Q_1 \circ Q_2 \end{aligned}$$

Quaternion Kinematics

Recall

$$P' = Q \circ P \circ \bar{Q}$$

Taking derivative yields

$$\begin{aligned}\dot{P}' &= \dot{Q} \circ P \circ \bar{Q} + Q \circ P \circ \dot{\bar{Q}} \\ &= \underbrace{\dot{Q} \circ \bar{Q} \circ P' \circ Q}_{P} \circ \bar{Q} + Q \circ \underbrace{\bar{Q} \circ P' \circ Q}_{P} \circ \dot{\bar{Q}} \\ &= \dot{Q} \circ \bar{Q} \circ P' + P' \circ Q \circ \dot{\bar{Q}}\end{aligned}$$

As $Q \circ \bar{Q} = 1$

Taking derivative yields

$$\dot{Q} \circ \bar{Q} + Q \circ \dot{\bar{Q}} = 0$$



$$\left. \begin{array}{l} Q \circ \dot{\bar{Q}} = -\dot{Q} \circ \bar{Q} \\ Q \circ \dot{\bar{Q}} = \overline{\dot{Q} \circ \bar{Q}} \end{array} \right\} \dot{Q} \circ \bar{Q} = 0 + a; Q \circ \dot{\bar{Q}} = 0 - a$$

Notice

$$\dot{P}' = (0 + a) \circ P' + P' \circ (0 - a)$$

Quaternion Kinematics

$$\begin{aligned}\dot{\mathbf{P}}' &= (0 + \mathbf{a}) \circ \mathbf{P}' + \mathbf{P}' \circ (0 - \mathbf{a}) \\ &= \begin{bmatrix} 0 \\ \mathbf{a} \end{bmatrix} \circ \begin{bmatrix} 0 \\ \mathbf{p}' \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{p}' \end{bmatrix} \circ \begin{bmatrix} 0 \\ -\mathbf{a} \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ \mathbf{a} \times \mathbf{p}' \end{bmatrix} - \mathbf{a} \cdot \mathbf{p}' + \begin{bmatrix} 0 \\ \mathbf{p}' \times (-\mathbf{a}) \end{bmatrix} - \mathbf{p}' \cdot (-\mathbf{a})\end{aligned}$$

$$\boxed{\begin{bmatrix} 0 \\ \dot{\mathbf{p}}' \end{bmatrix} = \begin{bmatrix} 0 \\ 2\mathbf{a} \times \mathbf{p}' \end{bmatrix}}$$

Recall $\dot{\mathbf{p}}' = \boldsymbol{\omega} \times \mathbf{p}'$, $\boldsymbol{\omega}$ is angular velocity vector represented in the **world** frame

Then $2\mathbf{a} = \boldsymbol{\omega} \Rightarrow \mathbf{a} = \frac{1}{2}\boldsymbol{\omega}$

$$\Rightarrow \dot{\mathbf{Q}} \circ \overline{\mathbf{Q}} = \begin{bmatrix} 0 \\ \mathbf{a} \end{bmatrix} = \frac{1}{2} \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}}_{\boldsymbol{\Omega}} \quad \Rightarrow \dot{\mathbf{Q}} = \frac{1}{2}\boldsymbol{\Omega} \circ \mathbf{Q}$$

$$\mathbf{Q} = \begin{bmatrix} q_0 \\ \mathbf{q} \end{bmatrix} \Rightarrow \dot{\mathbf{Q}} = \frac{1}{2}\boldsymbol{\Omega} \circ \mathbf{Q} = \mathcal{L}(\boldsymbol{\Omega})\mathbf{Q} = \frac{1}{2} \begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & \hat{\boldsymbol{\omega}} \end{bmatrix} \mathbf{Q}$$

Quaternion Kinematics

$$\dot{Q} = \frac{1}{2} \boldsymbol{\Omega} \circ Q$$

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}$$

$\boldsymbol{\omega}$ is angular velocity represented in the **world** frame

- What if $\boldsymbol{\omega}$ is represented in the **body** frame (i.e. $\boldsymbol{\omega}^b$)?

$$\boldsymbol{\Omega}^b = \begin{bmatrix} 0 \\ \boldsymbol{\omega}^b \end{bmatrix} \text{ is represented in body frame}$$

$$\Rightarrow \quad \boldsymbol{\Omega}^b = \overline{Q} \circ \boldsymbol{\Omega} \circ Q$$

$$\Rightarrow \quad \boxed{\dot{Q} = \frac{1}{2} Q \circ \boldsymbol{\Omega}^b}$$

$$Q = \begin{bmatrix} q_0 \\ \mathbf{q} \end{bmatrix}$$

$$\Rightarrow \dot{Q} = \frac{1}{2} Q \circ \boldsymbol{\Omega}^b = \mathcal{R}(\boldsymbol{\Omega}^b) Q = \frac{1}{2} \begin{bmatrix} 0 & -\boldsymbol{\omega}^{bT} \\ \boldsymbol{\omega}^b & -\widehat{\boldsymbol{\omega}^b} \end{bmatrix} Q$$

Unit Quaternion and Rotation

- Rotation to quaternion: $\mathbf{Q} = \begin{bmatrix} \cos \frac{\theta}{2} \\ \sin \frac{\theta}{2} \mathbf{u} \end{bmatrix}$, where $\mathbf{R} = e^{\hat{\mathbf{u}}\theta}$
- Quaternion to rotation
- $\mathbf{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(-q_0q_1 + q_2q_3) \\ 2(-q_0q_2 + q_1q_3) & 2(q_0q_1 + q_2q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$
- Rotating a vector: $\mathbf{P}' = \mathbf{Q} \circ \mathbf{P} \circ \bar{\mathbf{Q}}$
- Rotating a frame: $\mathbf{P}^b = \bar{\mathbf{Q}} \circ \mathbf{P} \circ \mathbf{Q}$
- Sequential rotation (extrinsic): $\mathbf{Q} = \mathbf{Q}_n \dots \circ \mathbf{Q}_2 \circ \mathbf{Q}_1$
- Sequential rotation (intrinsic): $\mathbf{Q} = \mathbf{Q}_1 \circ \mathbf{Q}_2 \dots \circ \mathbf{Q}_n$
- Kinematics under spatial frame: $\dot{\mathbf{Q}} = \frac{1}{2} \boldsymbol{\Omega} \circ \mathbf{Q} = \frac{1}{2} \begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & \hat{\boldsymbol{\omega}} \end{bmatrix} \mathbf{Q}$
- Kinematics under body frame: $\dot{\mathbf{Q}} = \frac{1}{2} \mathbf{Q} \circ \boldsymbol{\Omega} = \frac{1}{2} \begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & -\hat{\boldsymbol{\omega}} \end{bmatrix} \mathbf{Q}$