University of Zurich[UZH]

ETH zürich

ROBOTICS & PERCEPTION GROUP

Vision Algorithms for Mobile Robotics

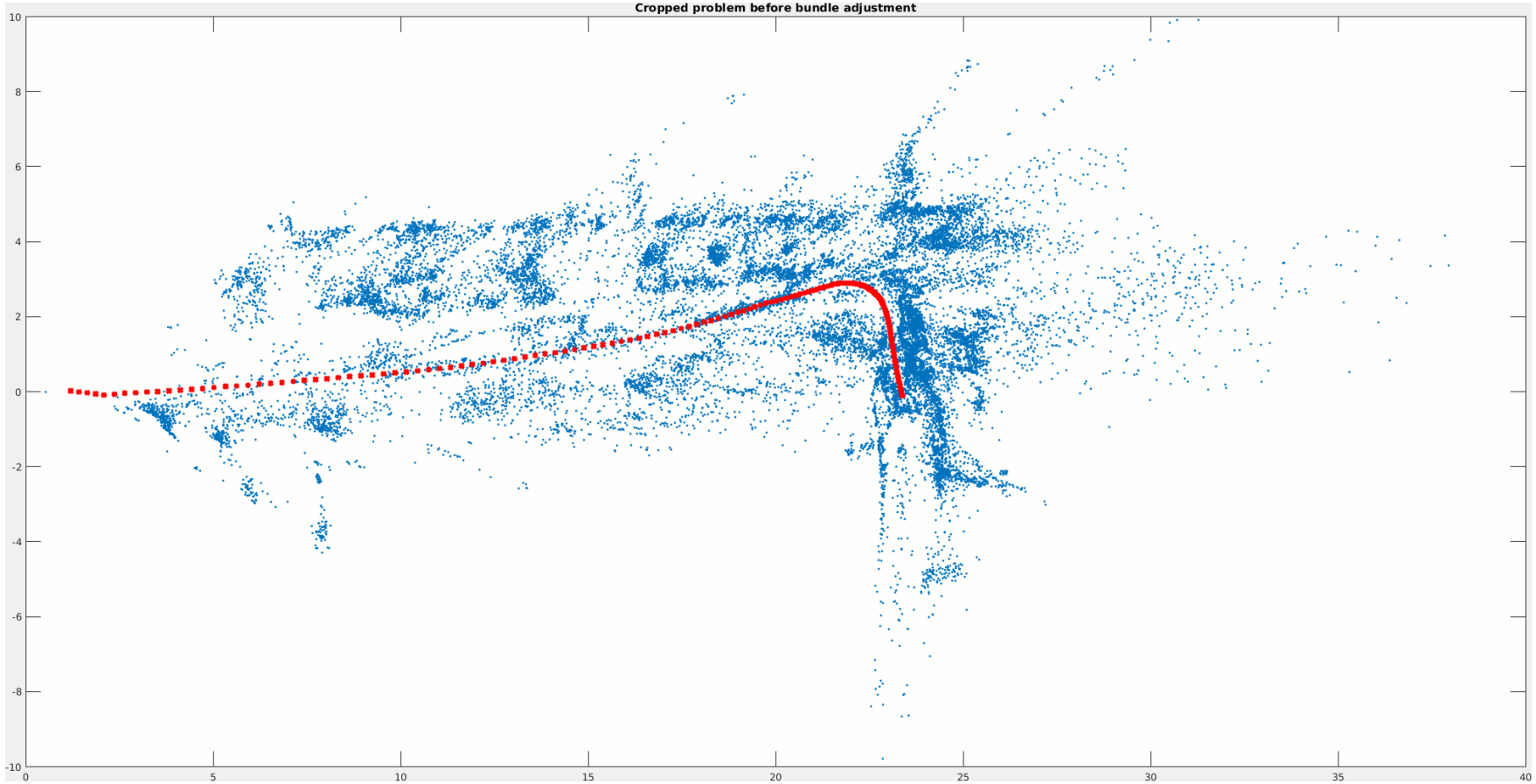
Lecture 13
Visual Inertial Fusion

Davide Scaramuzza

http://rpg.ifi.uzh.ch

Lab Exercise – This afternoon

Bundle Adjustment



Visual Inertial Odometry (VIO)

References:

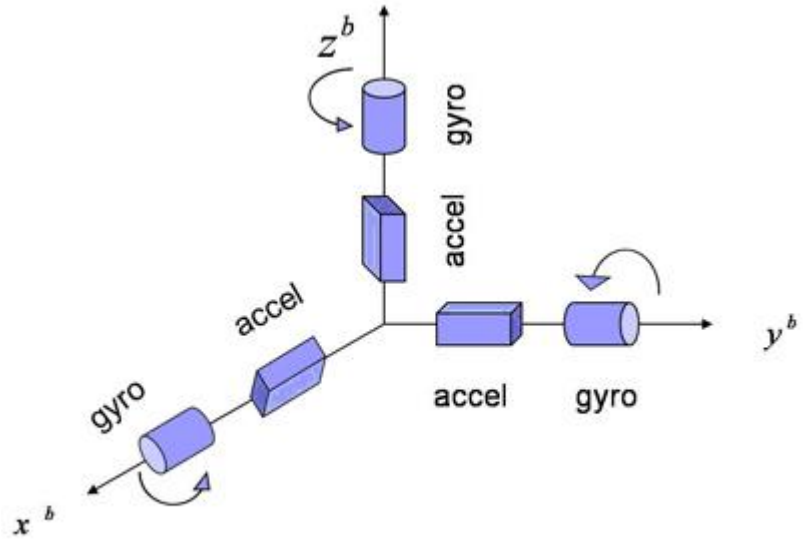
- Scaramuzza, Zhang, **Visual-Inertial Odometry of Aerial Robots**, Encyclopedia of Robotics, Springer, 2019, PDF.
- Huang, **Visual-inertial navigation: A concise review**, International conference on Robotics and Automation (ICRA), 2019. PDF.

Outline

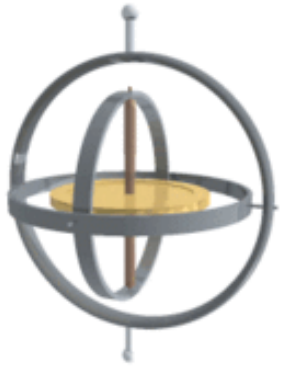
- What is an IMU and why do we need it?
- IMU model
- Visual Inertial Odometry
  - Closed-form solution
  - Non-linear optimization methods
  - Filtering methods
- Camera-IMU extrinsic calibration and Synchronization

What is an IMU?

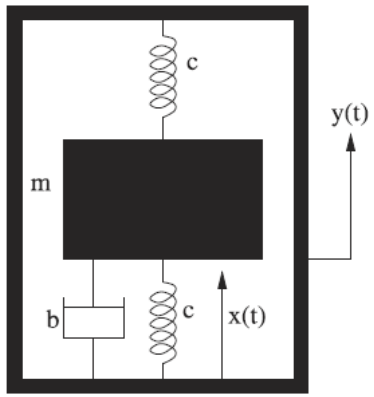
- **I**nertial **M**easurement **U**nit
  - Gyroscope: Angular velocity
  - Accelerometer: Linear Accelerations



Inertial Measurement Unit
3 accelerometers, 3 gyroscopes



Mechanical Gyroscope



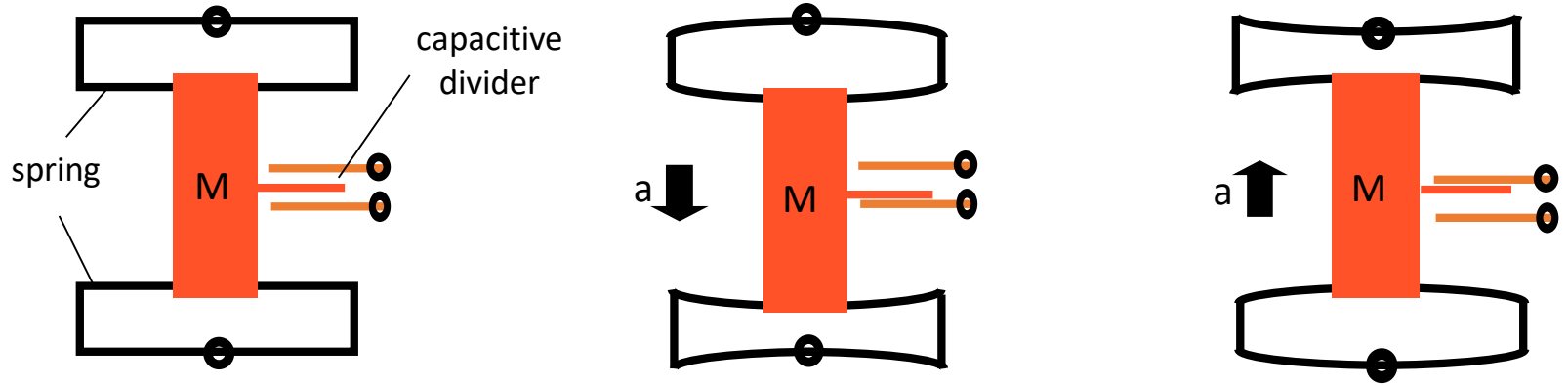
Mechanical Accelerometer

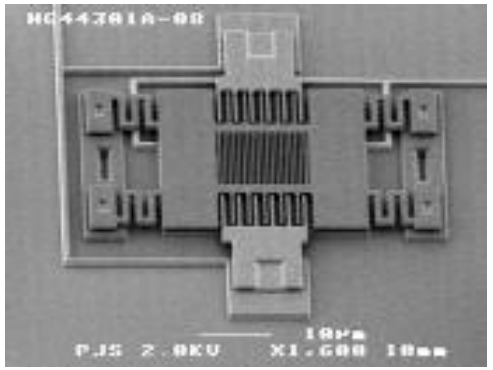
What is an IMU?

- Different categories
  - Mechanical ($100,000-1M)
  - Optical ($20,000-100k)
  - MEMS (from 1$ (phones) to 1,000$ (higher cost because they have a microchip running a Kalman filter))
- For small mobile robots & drones: MEMS IMU are mostly used
  - Cheap
  - Power efficient
  - Light weight and solid state

MEMS Accelerometer

A spring-like structure connects the device to a seismic mass vibrating in a capacitive divider. A capacitive divider converts the displacement of the seismic mass into an electric signal. Damping is created by the gas sealed in the device.





MEMS Gyroscopes

- MEMS gyroscopes measure the Coriolis forces acting on MEMS vibrating structures (tuning forks, vibrating wheels, or resonant solids)
- Their working principle is similar to the haltere of a fly
- Haltere are small structures of some two-winged insects, such as flies. They are flapped rapidly and function as gyroscopes, informing the insect about rotation of the body during flight.

Why do we need an IMU?

- Monocular vision is scale ambiguous
- Pure vision is not robust enough
  - Low texture
  - High Dynamic Range (HDR)
  - High speed motion

Motion blur

High Dynamic Range

Robustness is a critical issue: Tesla accident

> *"The autopilot sensors on the Model S failed to distinguish a white tractor-trailer crossing the highway against a bright sky. " [The Guardian]*

Why is IMU alone not enough?

- Pure IMU integration will lead to large drift (especially in cheap IMUs)
  - Will see later mathematically
  - Intuition
    - Double integration of acceleration to get position (e.g., 1D scenario: $x = x_0 + v_0(t_1 - t_0) + \iint_{t_0}^{t_1} a(t)dt^2$)
      If there is a constant bias in the acceleration, the error of position will be **proportional to t^2**
    - Integration of angular velocity to get orientation: if there is a bias in angular velocity, the error is **proportional to the time t**

GRADE/TIME	1 s	10 s	60 s	10 min	1 hr
Consumer	6 cm	6.5 m	400 m	200 km	39,000 km
Industrial	6 mm	0.7 m	40 m	20 km	3,900 km
Tactical	1 mm	8 cm	5 m	2 km	400 km
Navigation	<1 mm	1 mm	50 cm	100 m	10 km

Automotive, Smartphone, & Drone accelerometers

Table from Vectornav, one of the best IMU companies. Errors were computed assuming the device at rest:
https://www.vectornav.com/resources/ins-error-budget

Why visual inertial fusion?

- IMU and vision are complementary

Cameras

✓ Precise in slow motion
✓ More informative than an IMU (measures light energy from the environment)

X Limited output rate (~100 Hz)
X Scale ambiguity in monocular setup
X Lack of robustness to HDR and high speed

IMU

✓ Insensitive to motion blur, HDR, texture
✓ Can predict next feature position
✓ High output rate (~1,000 Hz)
✓ The higher the acceleration, the more accurate the IMU (due to higher signal to noise ratio)

X Large relative uncertainty when at low acceleration/angular velocity
X Ambiguity in gravity / acceleration (IMU measures the total acceleration)

- What cameras and IMU have in common: both estimate the pose incrementally (known as dead-reckoning), which suffers from drift over time. **Solution: loop detection and loop closure**

Outline

- What is an IMU and why do we need it?
- IMU model
- Visual Inertial Odometry
  - Closed-form solution
  - Non-linear optimization methods
  - Filtering methods
- Camera-IMU extrinsic calibration and Synchronization

IMU Measurement Model

- Measures angular velocity $\tilde{\omega}_B(t)$ and acceleration $\tilde{a}_B(t)$ in the body frame B:

$$\begin{aligned} \tilde{\omega}_B(t) &= \omega_B(t) + b^g(t) + n^g(t) \\ \tilde{a}_B(t) &= R_{BW}(t)(a_W(t) - g) + b^a(t) + n^a(t) \end{aligned}$$

IMU biases + noise

Raw IMU measurements

true $\boldsymbol{\omega}$ (in body frame) and true $\mathbf{a}$ (in world frame) to estimate

Notation:

- The superscript g stands for gyroscope and a for accelerometer
- R_{BW} is the rotation of the World frame W with respect to Body frame B
- The gravity g is expressed in the World frame
- Biases and noise are expressed in the body frame

IMU Noise Model

- Additive Gaussian white noise: $\mathbf{n}^g(t),\ \mathbf{n}^a(t)$
- Bias: $\mathbf{b}^g(t),\ \mathbf{b}^a(t)$
  - The gyroscope and accelerometer biases are considered slowly varying "constants". Their temporal fluctuation is modeled by saying that the derivative of the bias is a zero-mean Gaussian noise with standard deviation σ_b

$$\dot{\mathbf{b}}(t) = \sigma_b \mathbf{w}(t) \qquad \mathbf{w}(\mathrm{t}) \sim \mathbf{N}(\mathrm{o}, 1)$$

- Some facts about IMU biases:
- They change with **temperature** and **mechanical and atmospheric pressure**
- They **may change every time the IMU is started**
- Good news: **they can be estimated!** (see later)

Trawny, Roumeliotis, Indirect Kalman filter for 3D attitude estimation. Technical Report, University of Minnesota, 2005. PDF.
More info on the noise model: https://github.com/ethz-asl/kalibr/wiki/IMU-Noise-Model

IMU Integration Model

- The **IMU Integration Model** computes the position, orientation, and velocity of the IMU in the world frame. To do this, we must first compute the acceleration $a(t)$ in the world frame from the measured one $\tilde{a}(t)$ in the body frame (see Slide 13):

$$a(t) = R_{WB}(t)\big(\tilde{a}(t) - b(t)\big) + g$$

- The position p_k at time t_k can then be **predicted** from the position p_{k-1} at time t_{k-1} by integrating all the inertial measurements $\{\tilde{a}_j, \tilde{\omega}_j\}$ within that time interval:

$$p_k = p_{k-1} + v_{k-1}(t_k - t_{k-1}) + \iint_{t_{k-1}}^{t_k} R_{WB}(t)\big(\tilde{a}(t) - b(t)\big) + g)dt^2$$

- A similar expression can be obtained to predict the velocity v_k and orientation R_{WB} of the IMU in the world frame as functions of both $\tilde{a}_j$ and $\tilde{\omega}_j$

Trawny, Roumeliotis, Indirect Kalman filter for 3D attitude estimation. Technical Report, University of Minnesota, 2005. PDF.
More info on the noise model: https://github.com/ethz-asl/kalibr/wiki/IMU-Noise-Model

IMU Integration Model

For convenience, the **IMU Integration Model** is normally written as

$$\begin{pmatrix} p_k \\ q_k \\ v_k \end{pmatrix} = f\begin{pmatrix} p_{k-1} \\ q_{k-1}, u \\ v_{k-1} \end{pmatrix} \quad \text{or, more compactly:} \quad \boxed{x_k = f(x_{k-1}, u)}$$

where:

- $x = \begin{bmatrix} p \\ q \\ v \end{bmatrix}$ represents the IMU **state** comprising **position, orientation, and velocity**
- q is the IMU **orientation** $\boldsymbol{R_{WB}}$ (usually represented using quaternions)
- $u = \{\tilde{a}_j, \tilde{\omega}_j\}$ are the accelerometer and gyroscope measurements in the time interval $[t_{k-1}, t_k]$

Trawny, Roumeliotis, Indirect Kalman filter for 3D attitude estimation. Technical Report, University of Minnesota, 2005. PDF.
More info on the noise model: https://github.com/ethz-asl/kalibr/wiki/IMU-Noise-Model

Outline

- What is an IMU and why do we need it?
- IMU model
- Visual Inertial Odometry
  - Closed-form solution
  - Non-linear optimization methods
  - Filtering methods
- Camera-IMU extrinsic calibration and Synchronization

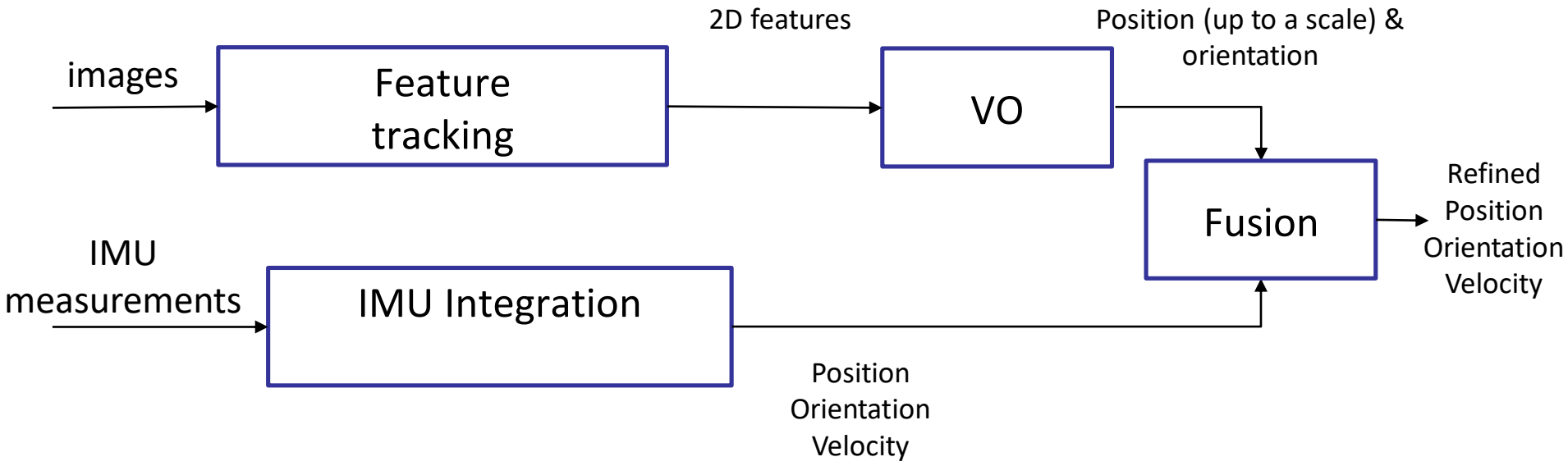
Visual Inertial Odometry

Different paradigms exist:

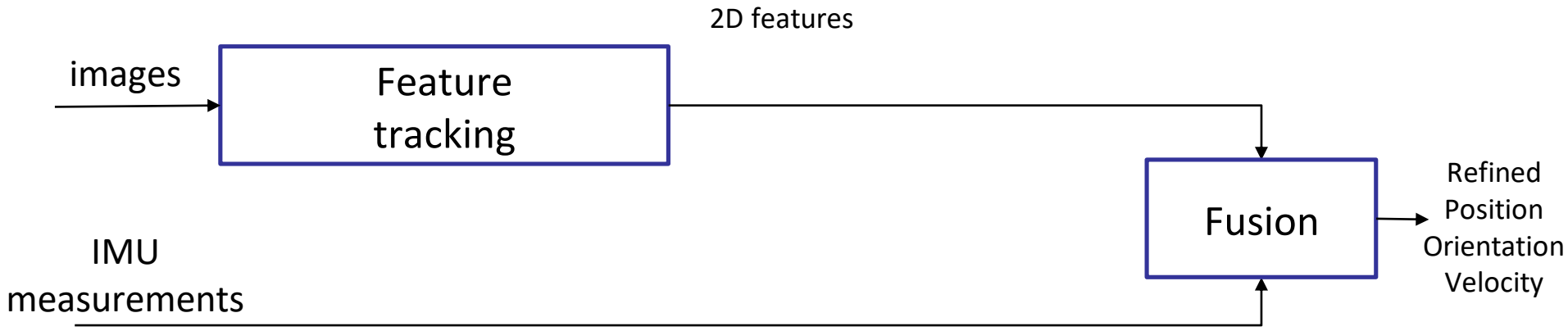
- **Loosely coupled**:
  - Treats VO and IMU as two separate (not coupled) black boxes
    - Each black box estimates pose and velocity from visual (up to a scale) and inertial data (absolute scale)
    - Easy to implement
    - Inaccurate. Should not be used
- **Tightly coupled**:
  - Makes use of the raw sensors' measurements:
    - 2D features
    - IMU readings
    - More accurate
    - More implementation effort

In this lecture, we will only see **tightly coupled approaches**

The Loosely Coupled Approach



The Tightly Coupled Approach



Filtering: Visual Inertial Formulation

- System states:

Tightly coupled: $\mathrm{X} = [p(t);\ q(t);\ v(t);\ \mathrm{b}^a(t);\ \mathrm{b}^g(t);\ L_1;\ L_2; \ldots;\ L_k]$

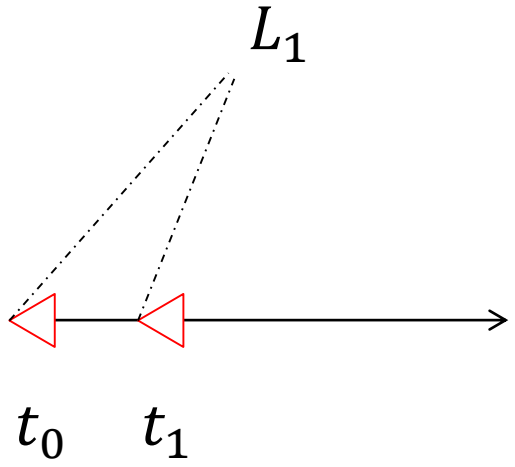
Loosely coupled: $\mathrm{X} = [p(t);\ q(t);\ v(t);\ \mathrm{b}^a(t);\ \mathrm{b}^g(t)]$

Corke, Lobo, Dias, *An Introduction to Inertial and Visual Sensing*, International Journal of Robotics Research (IJRR), 2017. PDF.

Outline

- What is an IMU and why do we need it?
- IMU model
- Visual Inertial Odometry
  - Closed-form solution
  - Non-linear optimization methods
  - Filtering methods
- Camera-IMU extrinsic calibration and Synchronization

Closed-form Solution (1D case)



- From the camera we can only get the absolute position x up to a unknown scale factor s, thus

$$x = s\tilde{x}$$

where $\tilde{x}$ is the relative motion estimated by the camera.

- From the IMU

$$x = x_0 + v_0(t_1 - t_0) + \iint_{t_0}^{t_1} a(t)dt^2$$

- By equating them

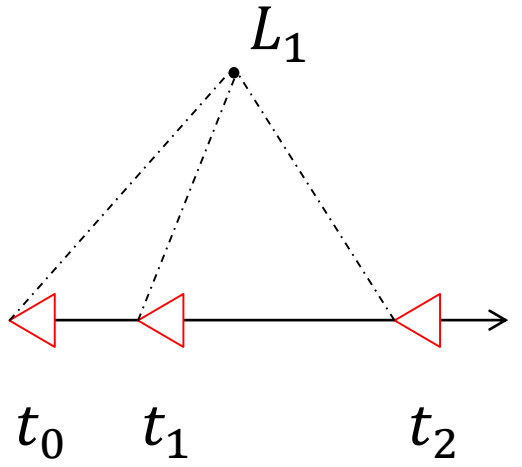
$$s\tilde{x} = x_0 + v_0(t_1 - t_0) + \iint_{t_0}^{t_1} a(t)dt^2$$

- As shown in [Martinelli'14], if we assume to know x_0 (usually we set it to 0), then, for 6DOF, **both s and v_0 can be determined in closed form** from a **single feature observation and 3 views**

Martinelli, *Closed-form solution of visual-inertial structure from motion*, International Journal of Computer Vision (IJCV), 2014. PDF.

Closed-form Solution (1D case)

$$\begin{cases} s\tilde{x}_1 = v_0(t_1 - t_0) + \iint_{t_0}^{t_1} a(t)dt^2 \\ s\tilde{x}_2 = v_0(t_2 - t_0) + \iint_{t_0}^{t_2} a(t)dt^2 \end{cases}$$



$$\begin{bmatrix} \tilde{x}_1 & (t_0 - t_1) \\ \tilde{x}_2 & (t_0 - t_2) \end{bmatrix} \begin{bmatrix} s \\ v_0 \end{bmatrix} = \begin{bmatrix} \iint_{t_0}^{t_1} a(t)dt^2 \\ \iint_{t_0}^{2} a(t)dt^2 \end{bmatrix}$$

Martinelli, *Closed-form solution of visual-inertial structure from motion*, International Journal of Computer Vision (IJCV), 2014. PDF.

Closed-form Solution (general case)

- Considers N feature observations and the full 6DOF case
- **Can be used to initialize filters and non-linear optimizers** (which always need an initialization point) [3]
- More complex to derive than the 1D case. But it also reaches a linear system of equations that can be solved using the pseudoinverse:

$$A\boldsymbol{X} = \boldsymbol{S}$$

$\boldsymbol{X}$ is the vector of unknowns:

- Absolute scale, s
- Initial velocity, v_0
- 3D Point distances (wrt the first camera)
- Direction of the gravity vector,
- Biases

$\boldsymbol{A}$ and $\boldsymbol{S}$ contain 2D feature coordinates, acceleration, and angular velocity measurements

$$A = \left[\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c} T_2 & S_2 & \Gamma_2 & \boldsymbol{\mu}_1^1 & 0_3 & 0_3 & -\boldsymbol{\mu}_2^1 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_{33} & 0_{33} & 0_{33} & \boldsymbol{\mu}_1^1 & -\boldsymbol{\mu}_1^2 & 0_3 & -\boldsymbol{\mu}_2^1 & \boldsymbol{\mu}_2^2 & 0_3 & 0_3 & 0_3 & 0_3 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0_{33} & 0_{33} & 0_{33} & \boldsymbol{\mu}_1^1 & 0_3 & -\boldsymbol{\mu}_1^N & -\boldsymbol{\mu}_2^1 & 0_3 & \boldsymbol{\mu}_2^N & 0_3 & 0_3 & 0_3 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ T_{n_i} & S_{n_i} & \Gamma_{n_i} & \boldsymbol{\mu}_1^1 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & -\boldsymbol{\mu}_{n_i}^1 & 0_3 & 0_3 \\ 0_{33} & 0_{33} & 0_{33} & \boldsymbol{\mu}_1^1 & -\boldsymbol{\mu}_1^2 & 0_3 & 0_3 & 0_3 & 0_3 & -\boldsymbol{\mu}_{n_i}^1 & \boldsymbol{\mu}_{n_i}^2 & 0_3 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0_{33} & 0_{33} & 0_{33} & \boldsymbol{\mu}_1^1 & 0_3 & -\boldsymbol{\mu}_1^N & 0_3 & 0_3 & 0_3 & -\boldsymbol{\mu}_{n_i}^1 & 0_3 & \boldsymbol{\mu}_{n_i}^N \end{array}\right]$$

[1] Martinelli, *Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination*, IEEE Transactions on Robotics (T-RO), 2012. PDF.
[2] Martinelli, *Closed-form solution of visual-inertial structure from motion*, International Journal of Computer Vision (IJCV), 2014. PDF.
[3] Kaiser, Martinelli, Fontana, Scaramuzza, Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation, IEEE Robotics and Automation Letters (R-AL), 2017. PDF.

Outline

- What is an IMU and why do we need it?
- IMU model
- Visual Inertial Odometry
  - Closed-form solution
  - Non-linear optimization methods
  - Filtering methods
- Camera-IMU extrinsic calibration and Synchronization

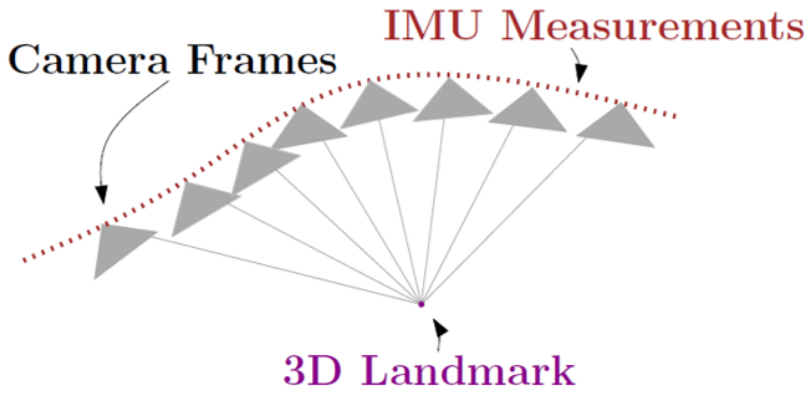
Non-linear Optimization Methods

VIO is solved as non-linear Least Square optimization problem over:

$$\{X, L, b^a, b^g\} = argmin_{\{X, L, b^a, b^g\}} \left\{ \sum_{k=1}^{N} \|f(x_{k-1}, u) - x_k\|^2_{\Lambda_k} + \sum_{k=1}^{N} \sum_{i=1}^{M} \left\|\pi(x_k, L_i) - z_{i_k}\right\|^2_{\Sigma_{i_k}} \right\}$$

IMU residuals

Reprojection residuals
(Bundle Adjustment term)



[1] Jung, Taylor, *Camera Trajectory Estimation using Inertial Sensor Measurements and Structure from Motion Results*, International Conference on Computer Vision and Pattern Recognition (CVPR), 2001. PDF.
[2] Sterlow, Singh, *Motion estimation from image and inertial measurements*, International Journal of Robotics Research (IJRR), 2004. PDF.

Non-linear Optimization Methods

VIO is solved as non-linear Least Square optimization problem over:

$$\{X, L, b^a, b^g\} = argmin_{\{X, L, b^a, b^g\}} \left\{ \sum_{k=1}^{N} \|f(x_{k-1}, u) - x_k\|^2_{\Lambda_k} + \sum_{k=1}^{N} \sum_{i=1}^{M} \left\|\pi(x_k, L_i) - z_{i_k}\right\|^2_{\Sigma_{i_k}} \right\}$$

IMU residuals — *Reprojection residuals* (Bundle Adjustment term)

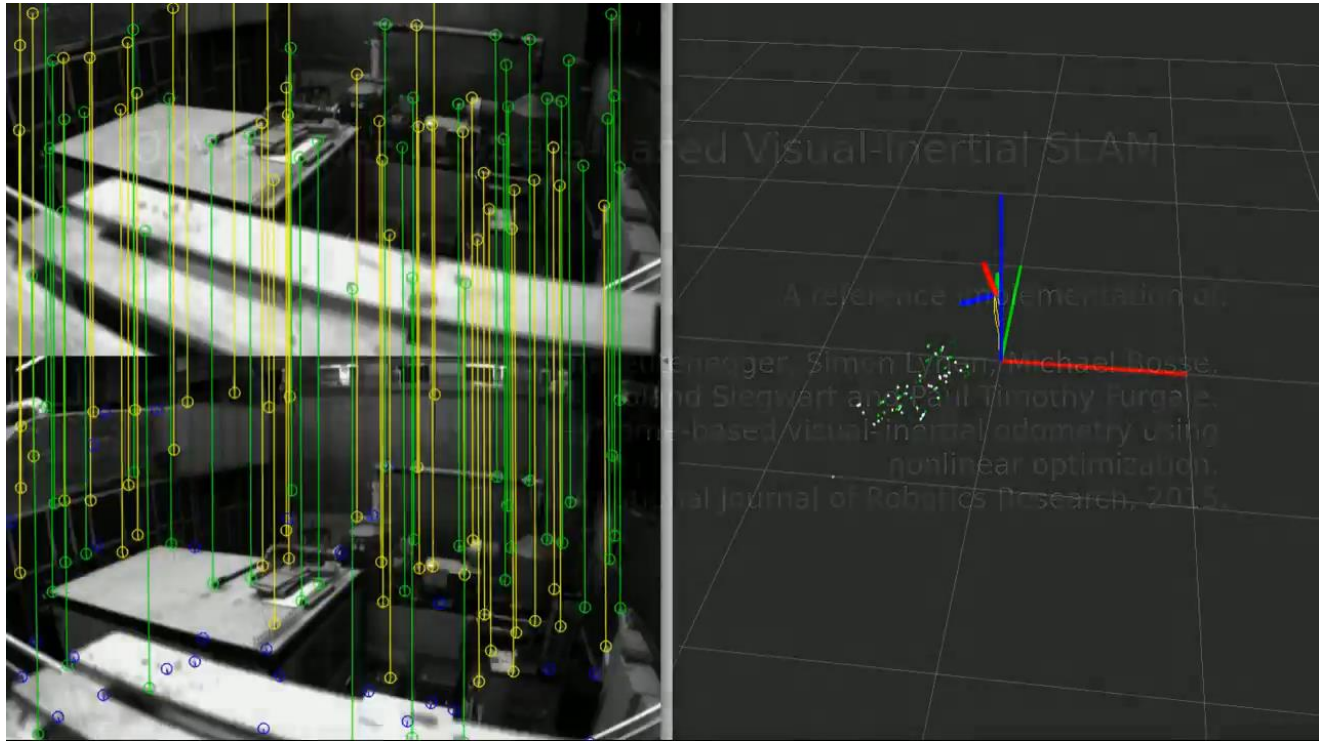
where

- $X = \{x_1, \dots x_N\}$: set of **robot state estimates** x_k (**position, velocity, orientation**) at frame times k
- $L = \{L_1, \dots, LM\}$: **3D landmarks**
- $f(x_{k-1}, u)$: pose prediction update from integration of IMU measurements $u = \{\tilde{a}_j, \tilde{\omega}_j\}$
- $\pi(x_k, l_i)$: measurement update from projection of landmark L_i onto camera frame I_k
- z_{i_k}: observed features
- Λ_k: **state covariance** from the IMU integration $f(x_{k-1}, u)$
- Σ_{i_k}: is the covariance from the noisy 2D feature measurements

[1] Jung, Taylor, *Camera Trajectory Estimation using Inertial Sensor Measurements and Structure from Motion Results*, International Conference on Computer Vision and Pattern Recognition (CVPR), 2001. PDF.
[2] Sterlow, Singh, *Motion estimation from image and inertial measurements*, International Journal of Robotics Research (IJRR), 2004. PDF.

Case Study 1: OKVIS

Because the complexity of the optimization is cubic with respect to the number of cameras poses and features, real-time operation becomes infeasible as the trajectory and the map grow over time, OKVIS proposed to only **optimize the current pose and a window of past keyframes**

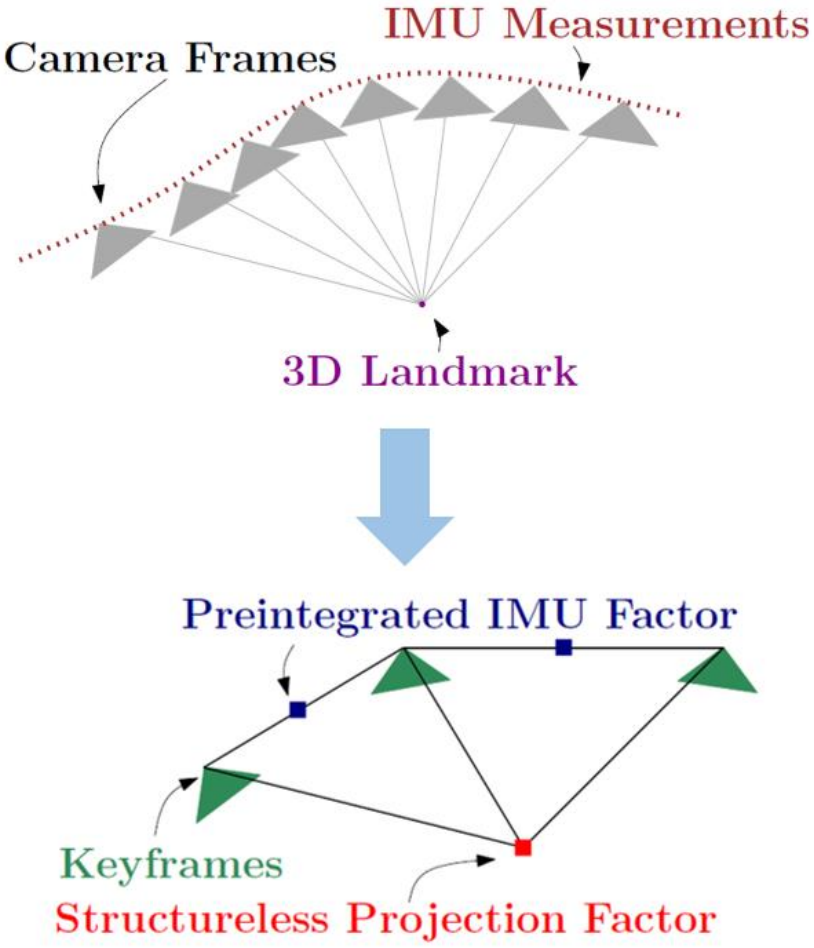


eutenegger, Lynen, Bosse, Siegwart, Furgale, *Keyframe-based visual–inertial odometry using nonlinear optimization*, International Journal of Robotics Research (IJRR), 2015. PDF.

Case Study 2: SVO+GTSAM

Solves the same optimization problem as OKVIS but:

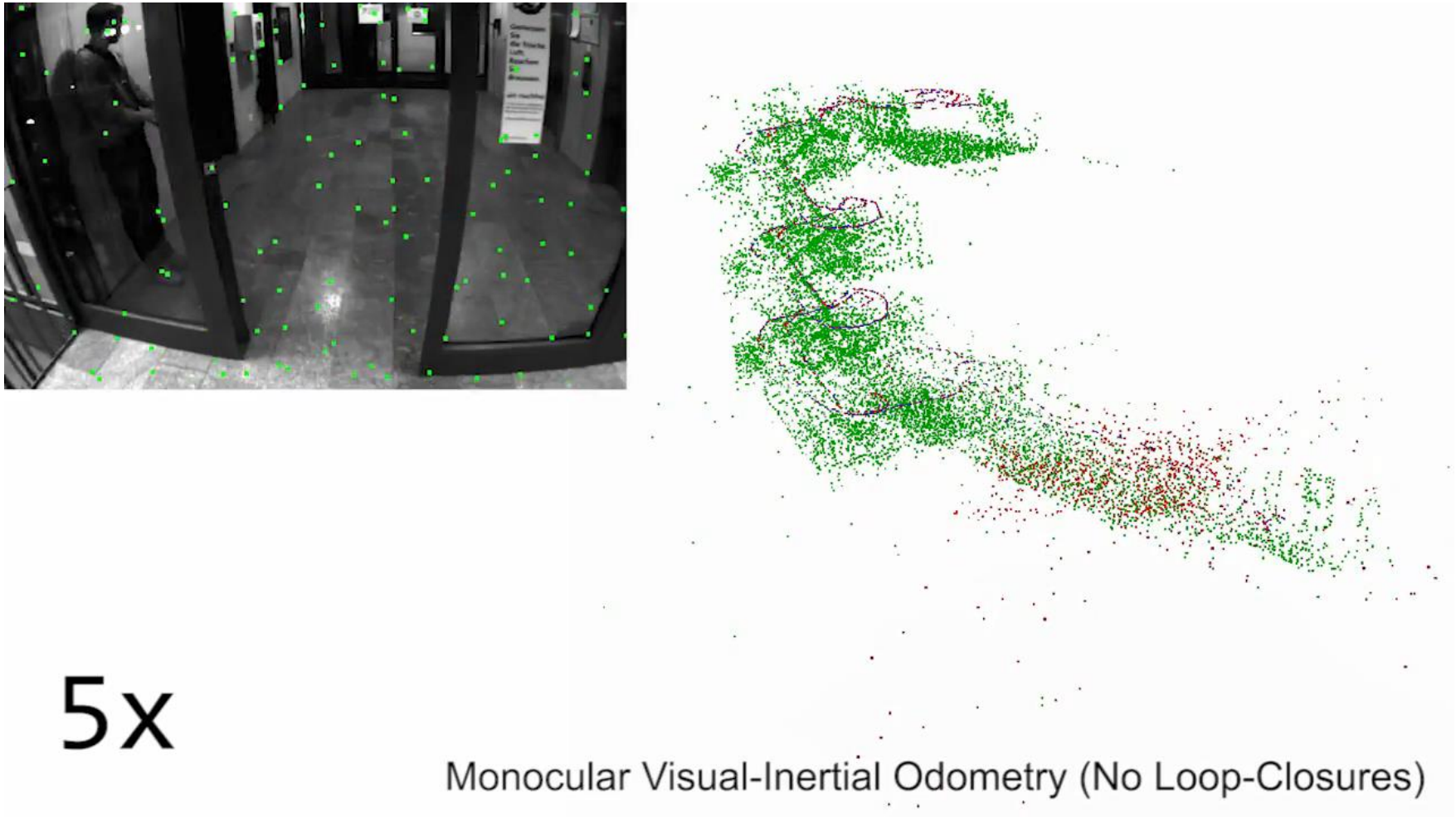
- **Keeps all the frames** (from the start to the end of the trajectory)
- To make the optimization efficient
  - **Marginalizes 3D landmarks** (rather than triangulating landmarks, only their visual bearing measurements are kept and the visual constraint are enforced by the **Epipolar Line Distance** (Lecture 08))
  - **pre-integrates the IMU** data between keyframes (see later)
- **Optimization solved using Factor Graphs** via GTSAM
  - Very fast because it only **optimizes** the **poses** that are **affected by a new observation**



SWISS MADE

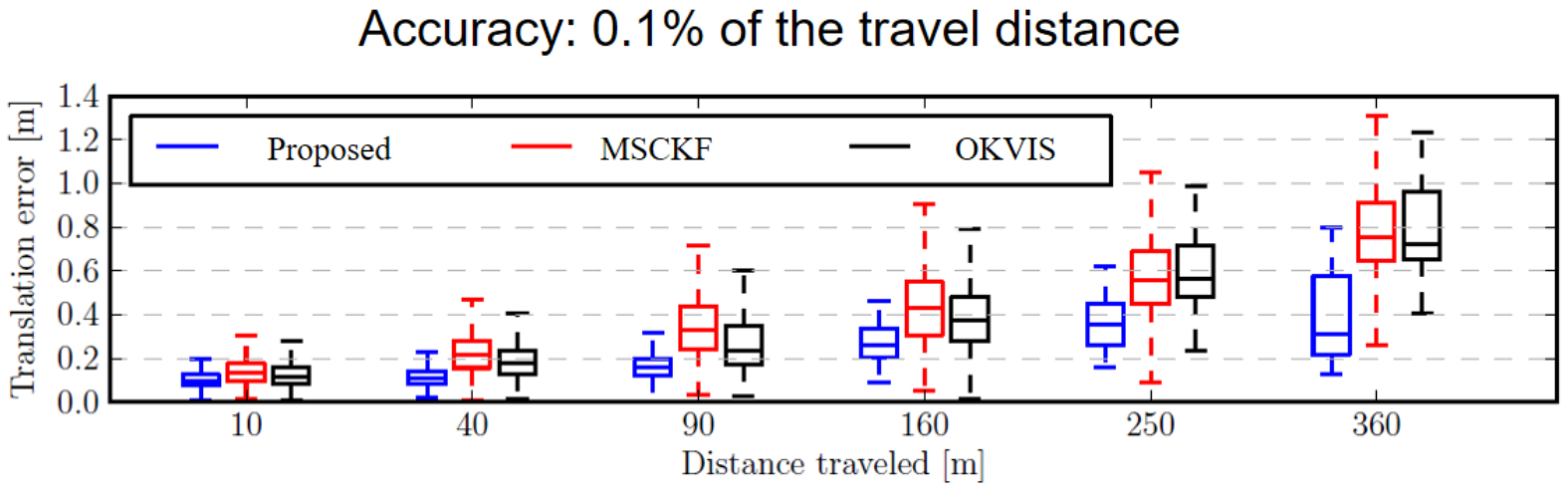
Forster, Carlone, Dellaert, Scaramuzza, *On-Manifold Preintegration for Real-Time Visual-Inertial Odometry*, IEEE Transactions on Robotics (T-RO), Feb. 2017, **Best Paper Award 2018**.

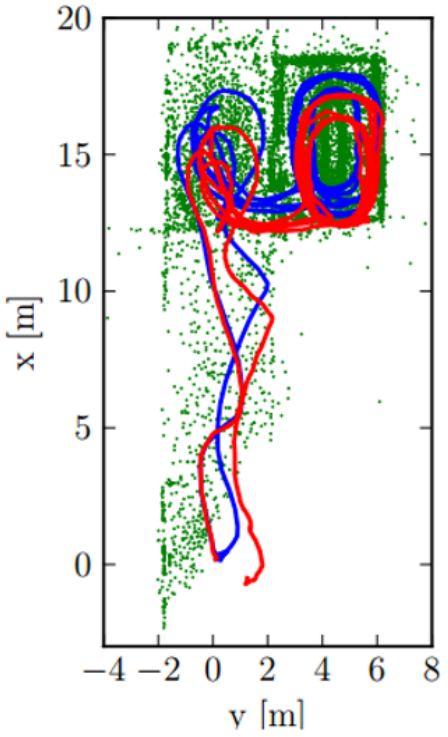
SVO+GTSAM



Forster, Carlone, Dellaert, Scaramuzza, *On-Manifold Preintegration for Real-Time Visual-Inertial Odometry*, IEEE Transactions on Robotics (T-RO), Feb. 2017, **Best Paper Award 2018**.

SVO+GTSAM





SWISS MADE

Forster, Carlone, Dellaert, Scaramuzza, *On-Manifold Preintegration for Real-Time Visual-Inertial Odometry*, IEEE Transactions on Robotics (T-RO), Feb. 2017, **Best Paper Award 2018**.

Problem with IMU integration

- The integration of IMU measurements, $f(x_{k-1}, u)$, from $k-1$ to k is related to the state estimation at time $k-1$
- During optimization, every time the linearization point at $k-1$ changes, the integration between $k-1$ and k must be re-evaluated, thus slowing down the optimization

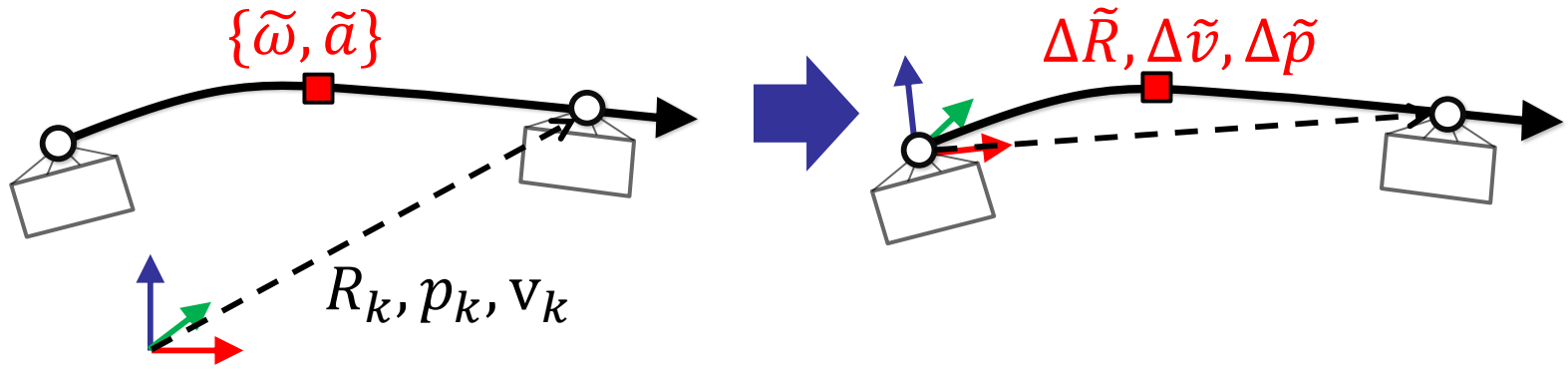
$$\{X, L, b^a, b^g\} = argmin_{\{X, L, b^a, b^g\}} \left\{ \sum_{k=1}^{N} \| f(x_{k-1}, u) - x_k \|^2_{\Lambda_k} + \sum_{k=1}^{N} \sum_{i=1}^{M} \left\| \pi(x_k, L_i) - z_{i_k} \right\|^2_{\Sigma_{i_k}} \right\}$$

IMU residuals — *Reprojection residuals* (Bundle Adjustment term)

- **Idea: Preintegration**
  - defines relative motion increments, expressed in body frame, which are independent on the global position, orientation, and velocity at k [1]
  - [2] uses this theory by leveraging the manifold structure of the rotation group SO(3)

[1] Lupton, Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions, IEEE Transactions on Robotics (T-RO), 2012. PDF.
[2] Forster, Carlone, Dellaert, Scaramuzza, *On-Manifold Preintegration for Real-Time Visual-Inertial Odometry*, IEEE Transactions on Robotics (T-RO), Feb. 2017, **Best Paper Award 2018**.

IMU Pre-Integration



Standard:
Evaluate **error in global frame**:

$$\boldsymbol{e}_R = \hat{R}(\tilde{\omega}, R_{k-1})^T R_k$$

$$\boldsymbol{e}_{\mathrm{v}} = \hat{\mathrm{v}}(\tilde{\omega}, \tilde{a}, \mathrm{v}_{k-1}) - \mathrm{v}_k$$

$$\boldsymbol{e}_p = \underbrace{\hat{p}(\tilde{\omega}, \tilde{a}, p_{k-1})}_{\text{Prediction}} - \underbrace{p_k}_{\text{Estimate}}$$

Repeat integration when previous state changes!

Preintegration:
Evaluate **relative errors**:

$$\boldsymbol{e}_R = \Delta\tilde{R}^T \Delta R$$

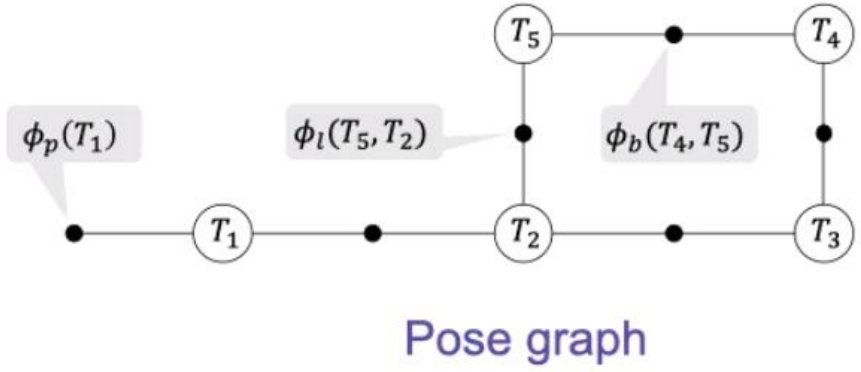
$$\boldsymbol{e}_{\mathrm{v}} = \Delta\tilde{\mathrm{v}} - \Delta\mathrm{v}$$

$$\boldsymbol{e}_p = \Delta\tilde{p} - \Delta p$$

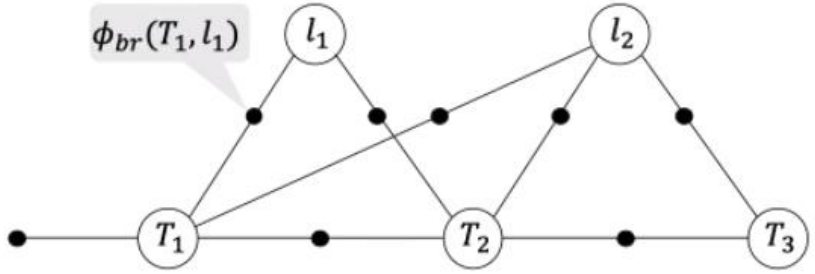
Preintegration of IMU deltas possible with **no initial condition required**.

GTSAM

- GTSAM [1] is a library that implements sensor fusion based on **factor graphs** and **Bayes trees**
- **Factor graphs** are a tool borrowed from machine learning [2] that
  - naturally capture the **spatial dependences** among different sensor measurements
  - can represent many robotics problems, like **pose graphs** and **SLAM**



Pose graph



SLAM

[1] Dellaert and Kaess, *Factor Graphs for Robot Perception*, Foundations and Trends in Robotics, 2017. PDF. Code.
[2] Kschischang, Frey, Loeliger, *Factor Graphs and the Sum-Product Algorithm*, Transactions on Information Theory, 2001. PDF.

GTSAM

- GTSAM uses factor graphs and Bayes trees to implement **incremental inference**

- A **Bayes tree** is a data structure, a directed tree, that visually represents the connectivity among the **past** robot states and measurements

- **Incremental inference** means that, when new robot states or measurements are added to the graph, only a **local part** of the tree is **updated**

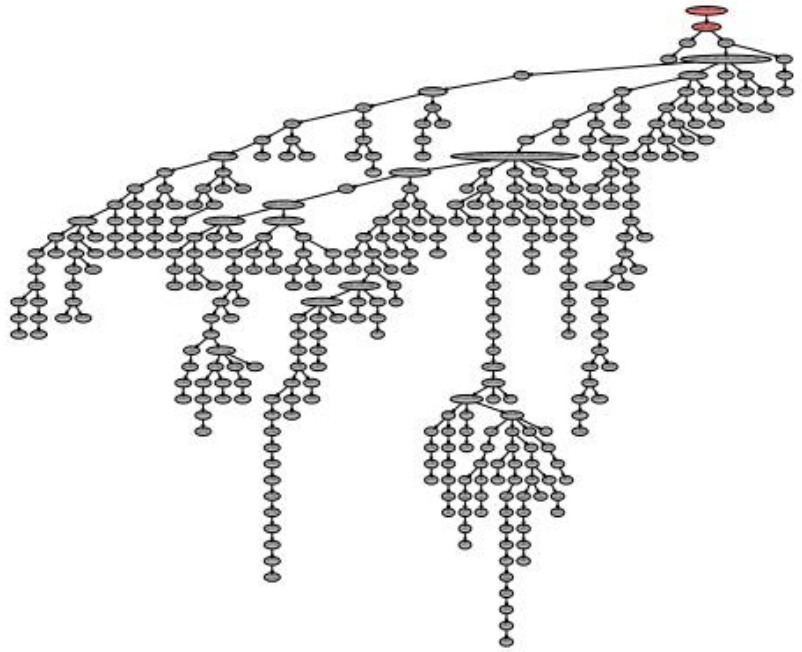


Figure 5.7: An example of the Bayes tree data structure for a small SLAM sequence. The incremental nonlinear least-squares estimation algorithm iSAM2 [109] is based on viewing incremental factorization as editing the graphical model corresponding to the posterior probability of the solution, the Bayes tree. As a robot explores the environment, new measurements often only affect small parts of the tree, and only those parts are re-calculated (shown in red).

[1] Kaess, Ranganathan, Dellaer, "iSAM: Incremental Smoothing and Mapping", IEEE Transactions on Robotics (T-RO), 2008. PDF.

Outline

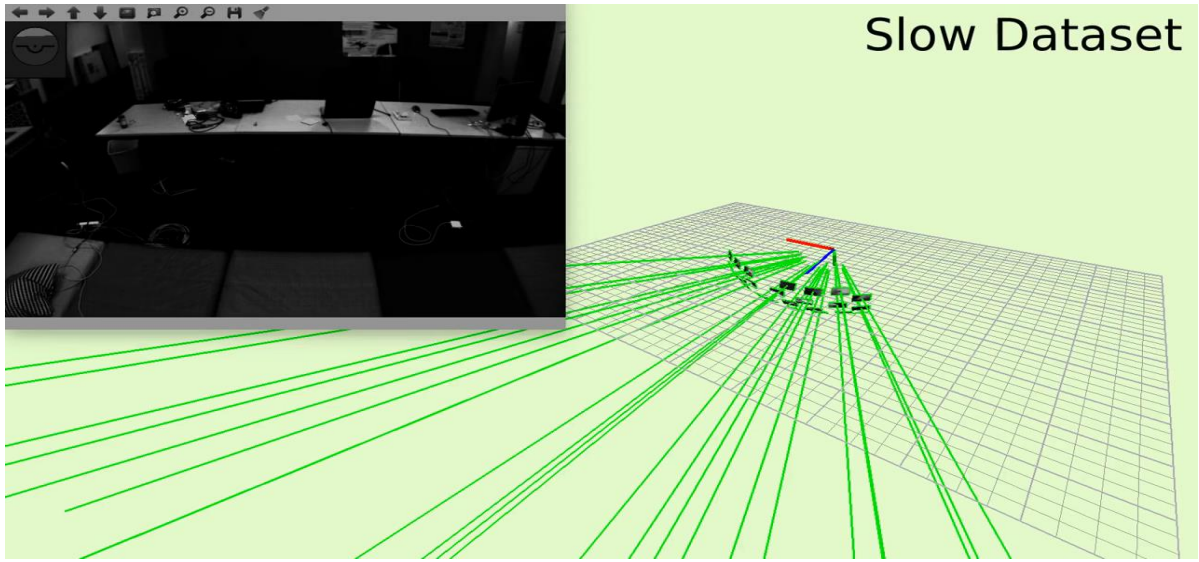
- What is an IMU and why do we need it?
- IMU model
- Visual Inertial Odometry
  - Closed-form solution
  - Non-linear optimization methods
  - Filtering methods
- Camera-IMU extrinsic calibration and Synchronization

Non-linear optimization vs. Filtering

Non-linear Optimization methods	Filtering methods
Optimize a **window** of multiple states (or all the states) using non-linear Least-Squares optimization	Solve the same problem by **running only one iteration of the optimization** function (e.g., using Extended Kalman Filter (EKF))
✓Multiple iterations (it re-linearizes at each iteration) ✓Acheives the highest accuracy ✓Slower (but fast with GTSAM)	×1 Linearization iteration only × Sensitive to linearization point ✓Fastest

Case Study 1: ROVIO

- EKF state: $\mathrm{X} = [p(t);\ q(t);\ v(t);\ \mathrm{b}^a(t);\ \mathrm{b}^g(t);\ L_1;\ L_2; \ldots;\ L_k]$
- Basic idea:
  1. **Prediction step**: predicts next position, velocity, orientation, and features using IMU integration model
  2. **Measurement update**: refines state by leveraging visual constraint (ROVIO minimizes the photometric error between corresponding points (alternative would be the reprojection error))



SWISS MADE

Bloesch, Burri, Omari, Hutter, Siegwart, *Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback*, International Journal of Robotics Research (IJRR), 2017. PDF. Code.

ROVIO: Problems

- **Complexity** of the EKF grows **quadratically** in the number of estimated landmarks
  - Thus, **max 20 landmarks** are tracked to allow real-time operation

- **Only updates the most recent state**

Case Study 2: MSCKF

- MSCKF (**Multi-State Constraint Kalman Filter**) updates **multiple past poses** $\{p_{C_1}, q_{C_1}, \dots, p_{C_N}, q_{C_N}\}$ in addition to the current state $\{p(t), q(t), v(t)\}$. State vector:

$$\mathrm{X} = \left[p(t);\ q(t);\ v(t);\ \mathrm{b}^a(t);\ \mathrm{b}^g(t);\ p_{C_1};\ q_{C_1};\ \dots;\ p_{C_N};\ q_{C_N}\right]$$

- **Prediction step:** same as ROVIO

- **Measurement update:**
  - Differently from ROVIO,
    - **landmark positions** are **not added to the state vector**, thus can run very fast independently of the number of features
    - Visual constraint is obtained from the **Epipolar Line Distance** (Lecture 08)

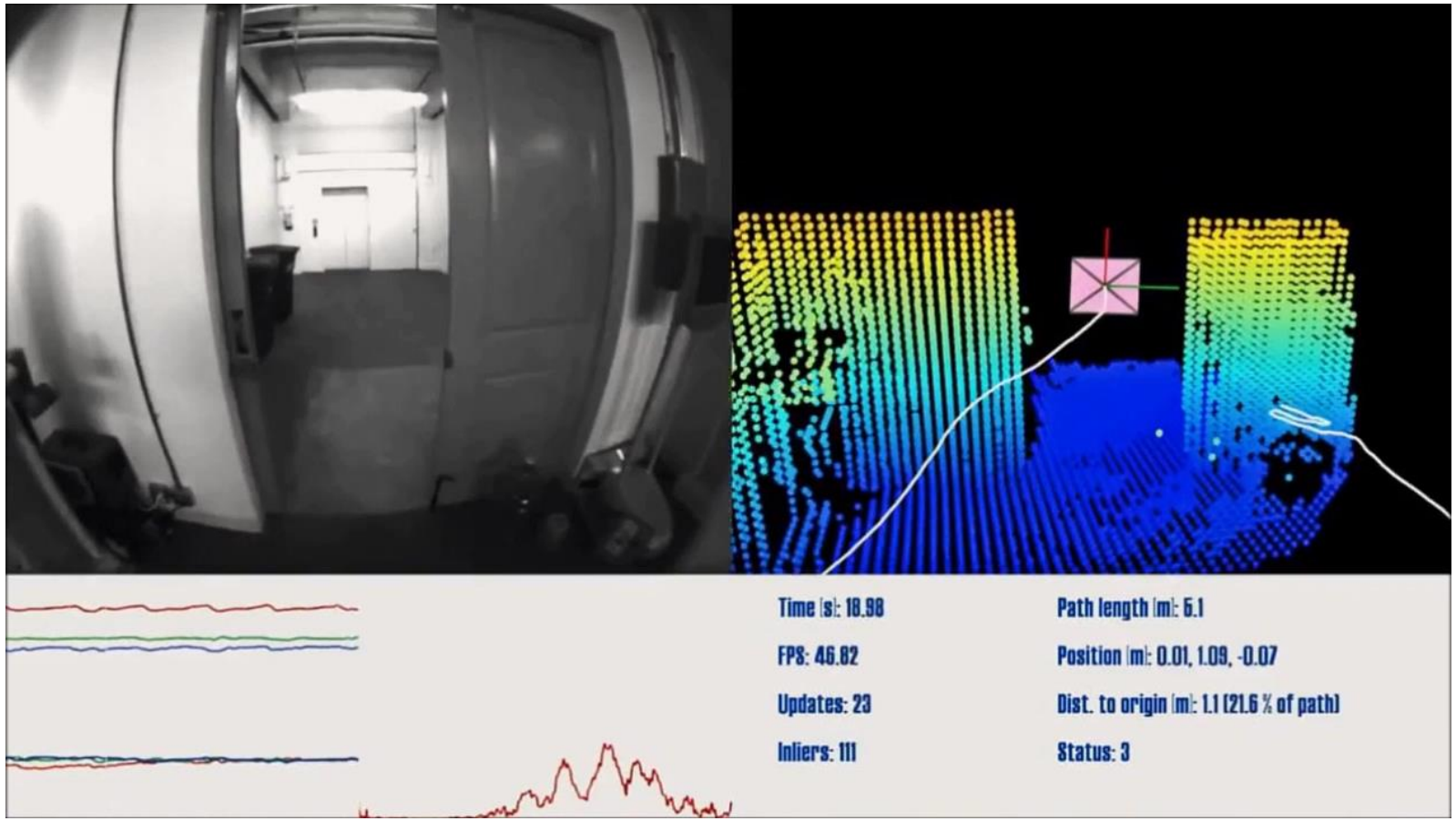
- Used in spacecraft landing (**NASA/JPL Moon and Mars landing**), **DJI drones**, **Google ARCore**

- Released open source within the OpenVins project: https://github.com/rpng/open_vins

[1] Mourikis, Roumeliotis, *A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation*, International Conference on Robotics and Automation (ICRA), 2007. PDF.

[2] Li, Mourikis, *High-precision, consistent EKF-based visual–inertial odometry*, International Journal of Robotics Research (IJRR), 2013. PDF.

MSCKF running in Google ARCore (former Google Tango)



[1] Mourikis, Roumeliotis, *A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation*, International Conference on Robotics and Automation (ICRA), 2007. PDF.

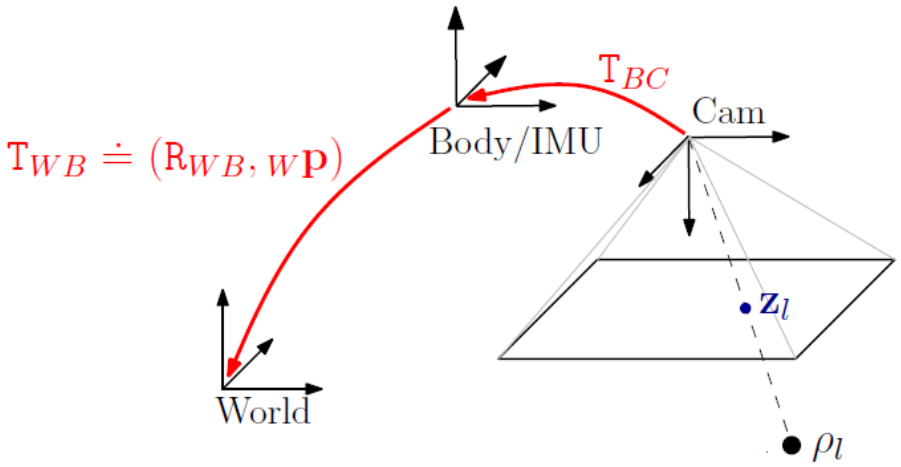
[2] Li, Mourikis, *High-precision, consistent EKF-based visual–inertial odometry*, International Journal of Robotics Research (IJRR), 2013. PDF.

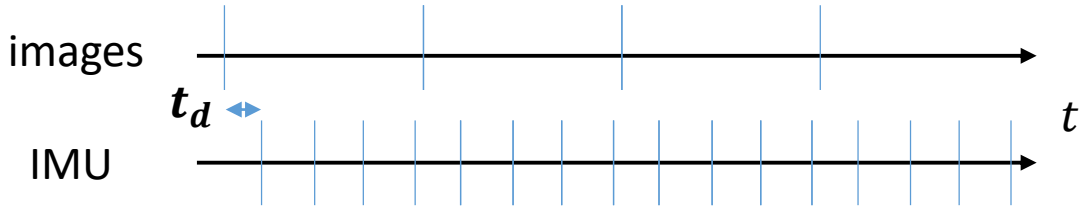
Outline

- What is an IMU and why do we need it?
- IMU model
- Visual Inertial Odometry
  - Closed-form solution
  - Non-linear optimization methods
  - Filtering methods
- Camera-IMU extrinsic calibration and Synchronization

Camera-IMU Calibration

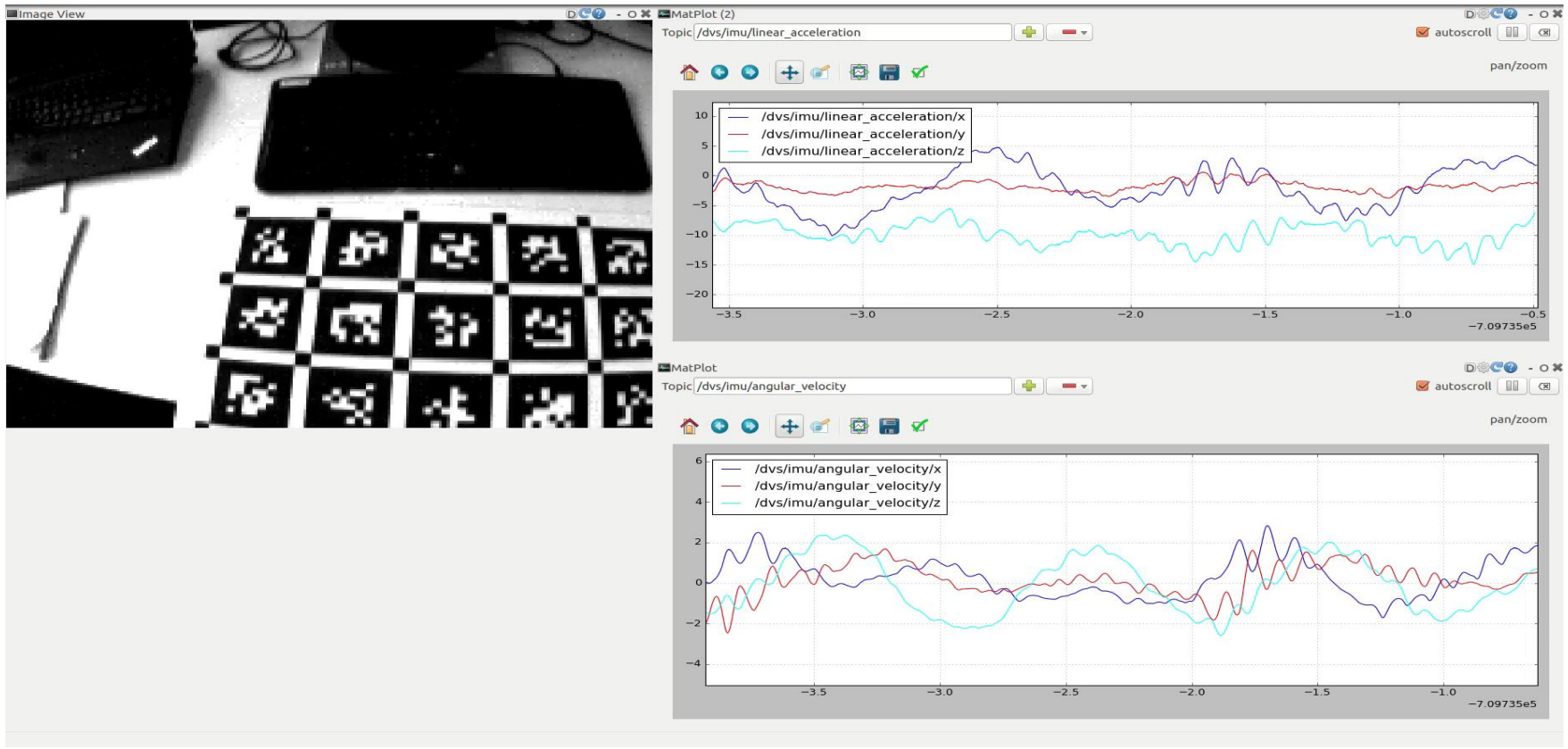
- **Goal**: estimate the **rigid-body transformation T_{BC}** and time **offset t_d between** the **camera** and the **IMU** caused by communication delays and the internal sensor delays (introduced by filters and logic).
- **Assumptions**: Camera and IMU rigidly attached. Camera intrinsically calibrated.
- **Data**:
  - Image points from calibration pattern (checkerboard or QR board)
  - IMU measurements: accelerometer $\{a_k\}$ and gyroscope $\{\omega_k\}$





Kalibr Toolbox

- Code: https://github.com/ethz-asl/kalibr/wiki/camera-imu-calibration



Furgale, Rehder, Siegwart, *Unified Temporal and Spatial Calibration for Multi-Sensor Systems*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013. PDF. Code.

Kalibr Toolbox

- Solves a non-linear Least Square optimization problem similar to that seen before but also optimizes over T_{BC}, t_d:

$$\{X, L, T_{BC}, t_d, b^a, b^g\} = argmin_{\{X, L, T_{BC}, t_d, b^a, b^g\}} \left\{ \sum_{k=1}^{N} \| f(x_{k-1}, u) - x_k \|^2_{\Lambda_k} + \sum_{k=1}^{N} \sum_{i=1}^{M} \left\| \pi(x_k, L_i) - z_{i_k} \right\|^2_{\Sigma_{i_k}} \right\}$$

IMU residuals

Reprojection residuals
(Bundle Adjustment term)

- Continuous-time modelling using splines for X
- Numerical solver: Levenberg-Marquardt

SWISS MADE

Furgale, Rehder, Siegwart, *Unified Temporal and Spatial Calibration for Multi-Sensor Systems*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013. PDF. Code.

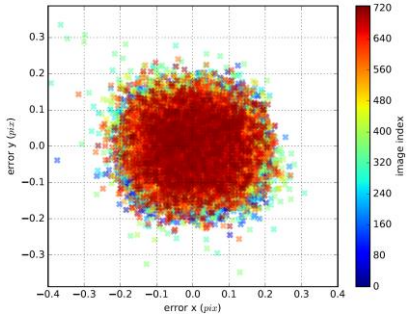
Kalibr Toolbox

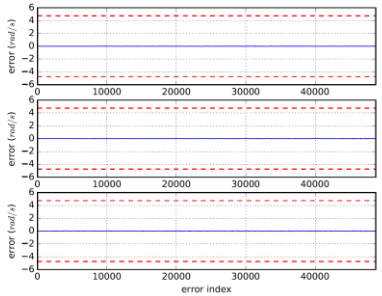
- Generates a report after optimizing the cost function.

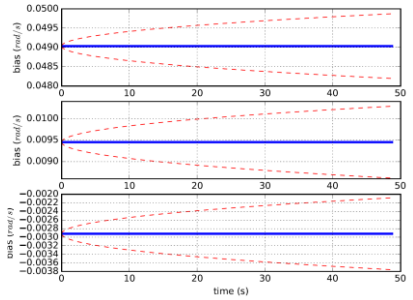
Residuals:
Reprojection error [px]: 0.0976 ± 0.051
Gyroscope error [rad/s]: 0.0167 ± 0.009
Accelerometer error [m/s^2]: 0.0595 ± 0.031

Transformation T_ci: (imu to cam):
[[0.99995526 -0.00934911 -0.00143776 0.00008436]
[0.00936458 0.99989388 0.01115983 0.00197427]
[0.00133327 -0.0111728 0.99993669 -0.05054946]
[0. 0. 0. 1.]]

Time shift (delay *d*)
cam0 to imu0: [s] (t_imu = t_cam + shift)
0.00270636270255





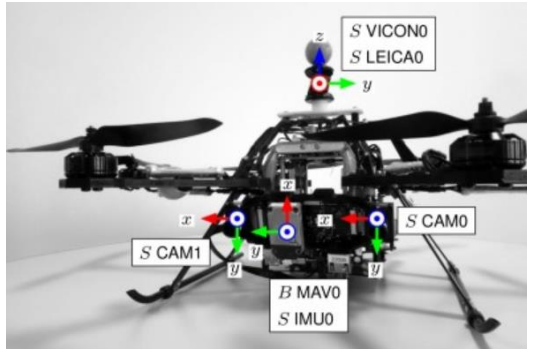


Furgale, Rehder, Siegwart, *Unified Temporal and Spatial Calibration for Multi-Sensor Systems*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013. PDF. Code.

Popular Datasets for VIO / VI-SLAM

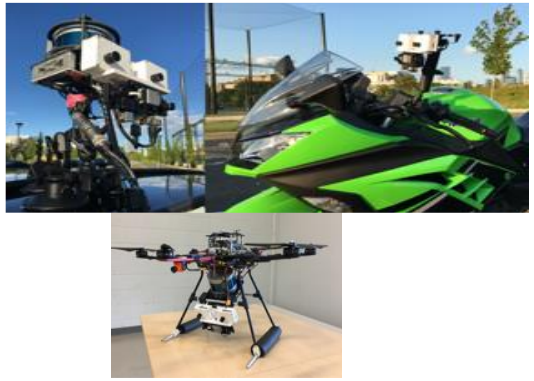
[EuRoC](#)

Drone with IMU and stereo camera



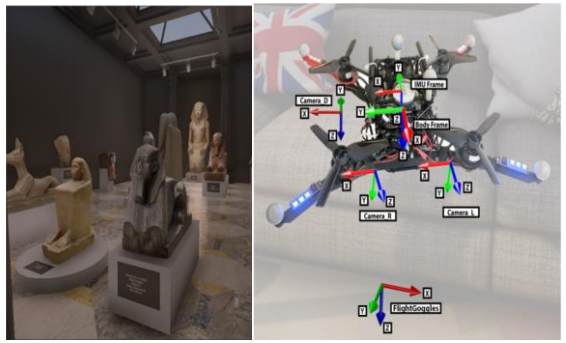
[MVSEC](#)

Drone, motorcycle, car with event camera, stereo camera, 3D lidar, GPS, IMU



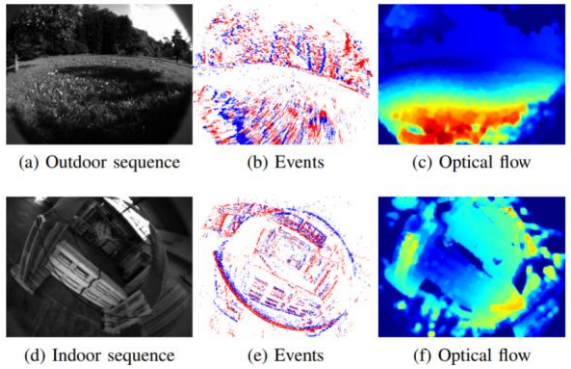
[Blackbird](#)

Drone flying fast with VR goggles (synthetic images and real dynamics + IMU)



[UZH FPV Drone Racing](#)

Drone flying fast with event camera, stereo camera, IMU



Understanding Check

Are you able to answer the following questions?

- Why is it recommended to use an IMU for Visual Odometry?
- Why not just an IMU, without a camera?
- What is the basic idea behind MEMS IMUs?
- What is the drift of a consumer IMU?
- What is the IMU measurement model? (formula)
- What causes the bias in an IMU?
- How do we model the bias?
- How do we integrate the acceleration to get the position (formula)?
- What is the definition of loosely coupled and tightly coupled visual inertial fusion?
- How does non-linear optimization-based visual inertial odometry? Can you write down the cost function and illustrate its meaning?