University of Zurich UZH | ETH zürich

Institute of Informatics – Institute of Neuroinformatics

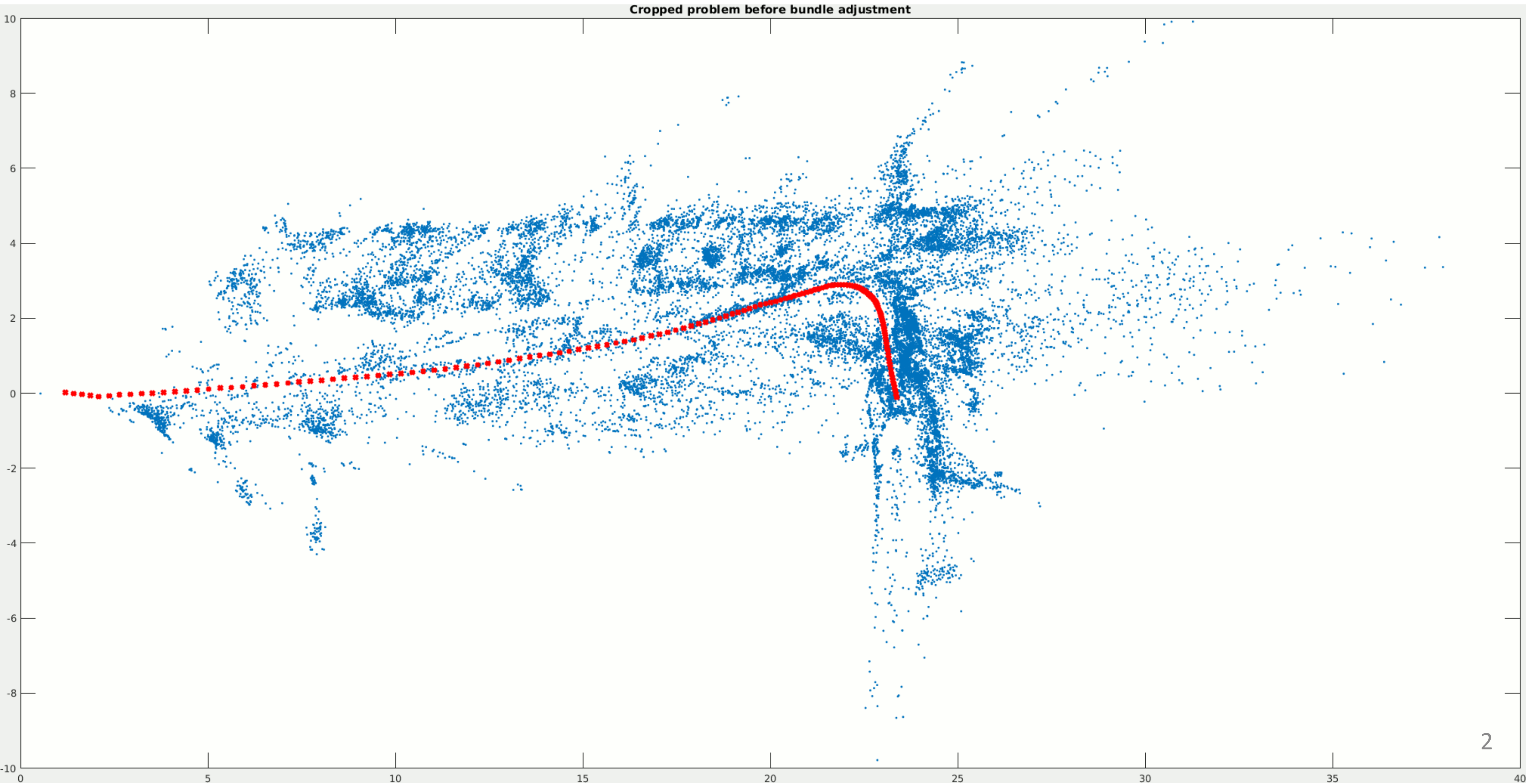
ROBOTICS & PERCEPTION GROUP

Lecture 13
Visual Inertial Fusion
(advanced)

Davide Scaramuzza

Lab Exercise 6 - Today

- Room ETH HG E 1.1 from 13:15 to 15:00
- Work description: Bundle Adjustment



Recall: VO Working Principle

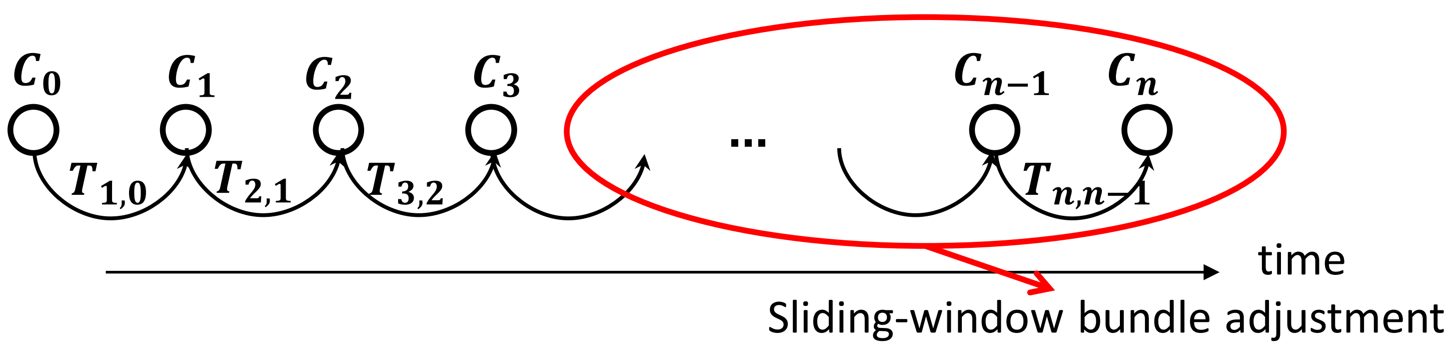
1. Compute the relative motion T_k from images I_{k-1} to image I_k

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}$$

2. Concatenate them to recover the full trajectory

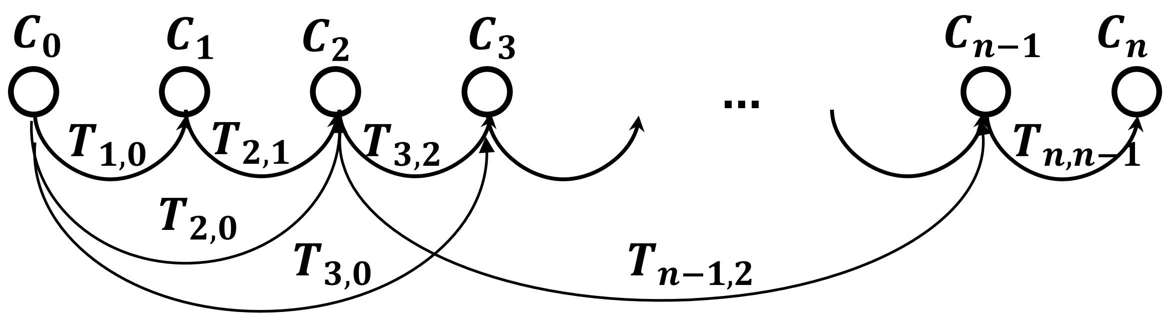
$$C_n = C_{n-1} T_n$$

3. An optimization over the last *m* poses can be done to refine locally the trajectory (Pose-Graph or Bundle Adjustment)



Pose-Graph Optimization

- So far we assumed that the transformations are between consecutive frames

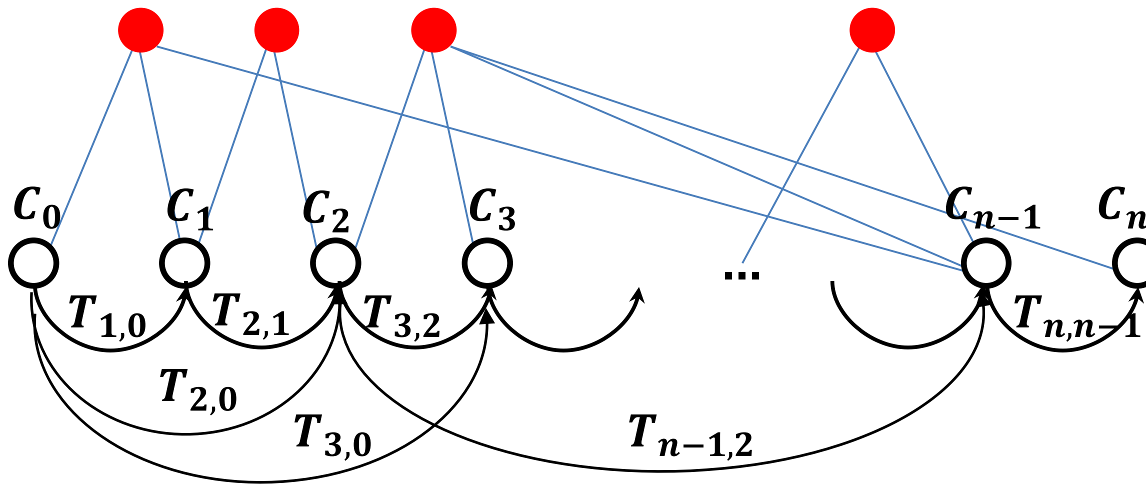


- Transformations can be computed also between non-adjacent frames T_{ij} (e.g., when features from previous keyframes are still observed). They can be used as additional constraints to improve cameras poses by minimizing the following:

$$C_k = argmin_{C_k}, \sum_i \sum_j \left\| C_i - C_j T_{ij} \right\|^2$$

- For efficiency, only the last m keyframes are used
- Gauss-Newton or Levenberg-Marquadt are typically used to minimize it. For large graphs, efficient open-source tools: **g2o**, **GTSAM**, **SLAM++**, **Google Ceres**

Bundle Adjustment (BA)



- Similar to pose-graph optimization but it also optimizes 3D points

$$X^i, C_k = argmin_{X^i, C_k,} \sum_i \sum_k \rho\left(p_k^i - \pi\left(X^i, C_k\right)\right)$$

- $\rho_H()$ is a robust cost function (e.g., Huber or Tukey cost) to penalize wrong matches
- In order to not get stuck in local minima, the initialization should be close to the minimum
- Gauss-Newton or Levenberg-Marquadt are typically used to minimize it. For large graphs, efficient open-source tools: **g2o**, **GTSAM**, **SLAM++**, **Google Ceres**

Huber and Tukey Norms

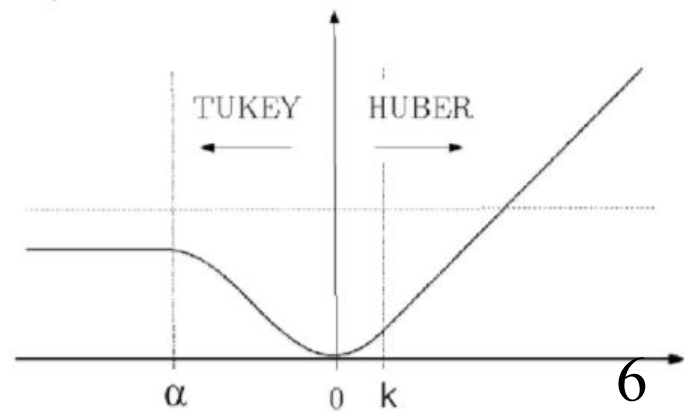
Goal: penalize the influence of wrong matches (i.e., high reprojection error)

- **Huber norm**:

$$\rho(x) = \begin{cases} x^2 & \text{if } |x| \leq k \\ k(2|x| - k) & \text{if } |x| \geq k \end{cases}$$

- **Tukey norm**:

$$\rho(x) = \begin{cases} \alpha^2 & \text{if } |x| \geq \alpha \\ \alpha^2 \left(1 - \left(1 - \left(\frac{x}{\alpha}\right)^2\right)^3\right) & \text{if } |x| \leq \alpha \end{cases}$$



Bundle Adjustment vs Pose-graph Optimization

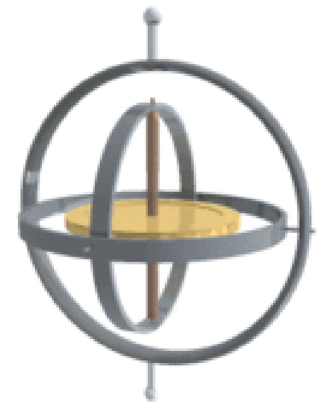
- BA is **more precise** than pose-graph optimization because it adds additional constraints (*landmark constraints*)
- But **more costly**: $O\left((qM + lN)^3\right)$ with M and N being the number of points and cameras poses and q and l the number of parameters for points and camera poses. Workarounds:
  - A **small window size** limits the number of parameters for the optimization and thus makes real-time bundle adjustment possible.
  - It is possible to reduce the computational complexity by just optimizing over the camera parameters and keeping the 3-D landmarks fixed, e.g., (**motion-only BA**)

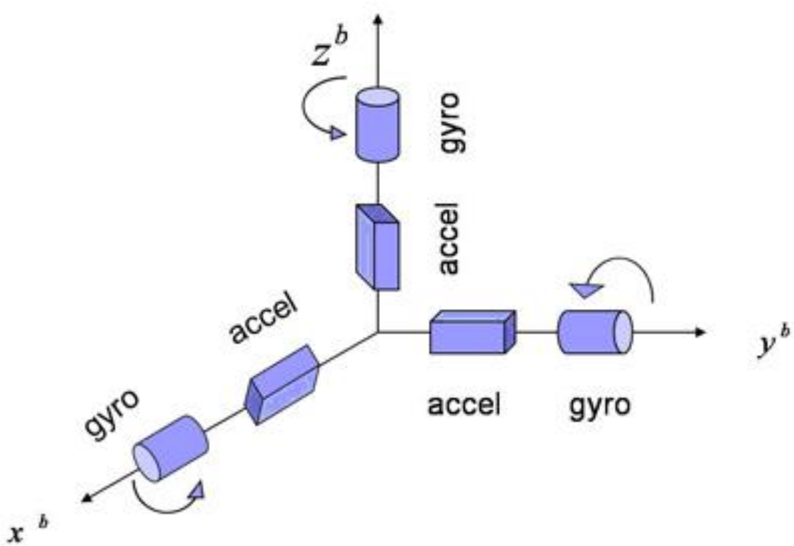
Outline

- Introduction
- IMU model and Camera-IMU system
- Different paradigms
  - Closed-form solution
  - Filtering approaches
  - Maximum a posteriori estimation (non linear optimizers)
    - Fixed-lag Smoothing (aka sliding window estimators)
    - Full smoothing methods
- Camera-IMU extrinsic calibration and Synchronization

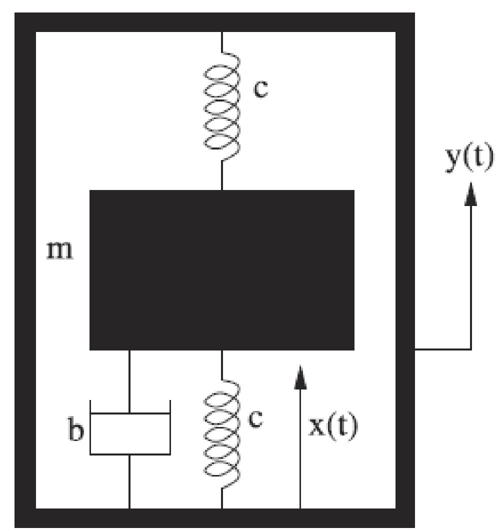
What is an IMU?

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





Inertial Measurement Unit
3 accelerometers, 3 gyroscopes



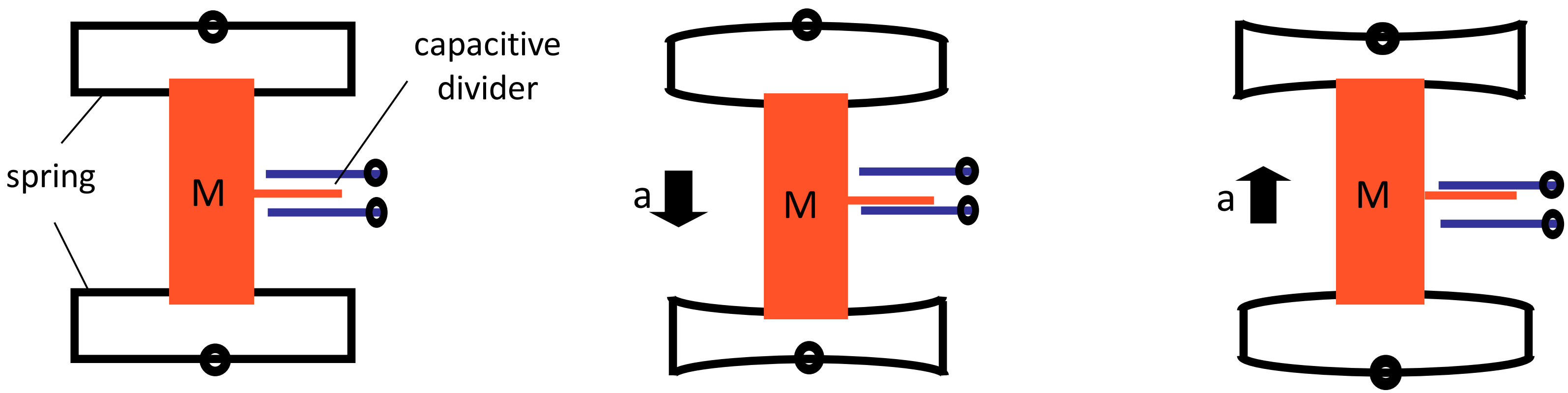
What is an IMU?

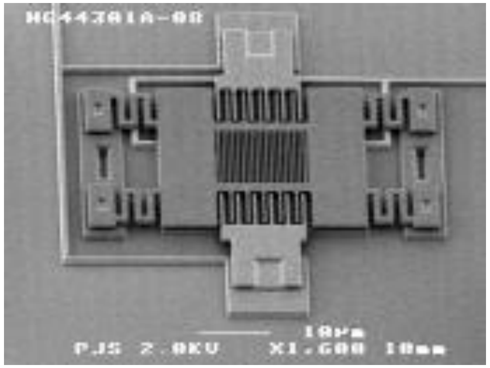
- Different categories
  - Mechanical ($100,000)
  - Optical ($20,000)
  - MEMS (from 1$ (phones) to 1,000$)
  - ....
- For mobile robots: MEMS IMU
  - Cheap
  - Power efficient
  - Light weight and solid state

10

MEMS Accelerometer

A spring-like structure connects the device to a seismic mass vibrating in a capacity devider. 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 IMU?

- Monocular vision is scale ambiguous.
- Pure vision is not robust enough
  - Low texture
  - High dynamic range
  - High speed motion

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. "

Why vision?

- Pure IMU integration will lead to large drift (especially cheap IMUs)
  - Will see later mathematically
  - Intuition
    - Integration of angular velocity to get orientation: error **proportional to t**
    - Double integration of acceleration to get position: if there is a bias in acceleration, the error of position is **proportional to t^2**
    - Worse, the actually position error also depends on the error of orientation.

Grade	Accelerometer Bias Error [mg]		Horizontal Position Error [m] 1s	10s	60s	1hr
Navigation	0.025		0.13 mm	12 mm	0.44 m	1.6 km
Tactical	0.3		1.5 mm	150 mm	5.3 m	19 km
Industrial	3		15 mm	1.5 m	53 m	190 km
Automotive	125		620 mm	60 m	2.2 km	7900 km

Smartphone accelerometers

http://www.vectornav.com/support/library/imu-and-ins

Why visual inertial fusion?

IMU and vision are complementary

Cameras

- ✓ Precise in slow motion
- ✓ Rich information for other purposes

- x Limited output rate (~100 Hz)
- x Scale ambiguity in monocular setup.
- x Lack of robustness

IMU

- ✓ Robust
- ✓ High output rate (~1,000 Hz)
- ✓ Accurate at high acceleration

- x Large relative uncertainty when at low acceleration/angular velocity
- x Ambiguity in gravity / acceleration

In common: state estimation based on visual or/and inertial sensor is dead-reckoning, which suffers from drifting over time.
(solution: loop detection and loop closure)

Outline

- Introduction
- IMU model and Camera-IMU system
- Different paradigms
  - Closed-form solution
  - Filtering approaches
  - Maximum a posteriori estimation (non linear optimizers)
    - Fixed-lag Smoothing (aka sliding window estimators)
    - Full smoothing methods
- Camera-IMU extrinsic calibration and Synchronization

IMU model: Measurement Model

- Measures angular velocity and acceleration in the body frame:

$$
\begin{aligned}
{}_{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{WB}}(t) &= {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{WB}}(t) + \mathbf{b}^{g}(t) + \mathbf{n}^{g}(t) \\
{}_{\mathrm{B}}\tilde{\mathbf{a}}_{\mathrm{WB}}(t) &= \mathbf{R}_{\mathrm{BW}}(t)\left({}_{\mathrm{W}}\mathbf{a}_{\mathrm{WB}}(t) - {}_{\mathrm{W}}\mathbf{g}\right) + \mathbf{b}^{a}(t) + \mathbf{n}^{a}(t)
\end{aligned}
$$

measurements

noise

where the superscript ^{g} stands for Gyroscope and ^{a} for Accelerometer

Notations:

- Left subscript: reference frame in which the quantity is expressed
- Right subscript {*Q*}{Frame1}{Frame2}: *Q* of Frame2 with respect to Frame1
- Noises are all in the body frame

IMU model: Noise Property

- Additive Gaussian white noise: $\mathbf{n}^g(t),\ \mathbf{n}^a(t)$

$$E\left[n(t)\right]=0$$

$$E\left[n(t_1)n(t_2)\right]=\sigma^2\delta(t_1-t_2)$$

$$n[k]=\sigma_d w[k]$$

$$w[k]\sim N(0,1)$$

$$\sigma_d=\sigma/\sqrt{\Delta \mathrm{t}}$$

- Bias: $\mathbf{b}^g(t),\ \mathbf{b}^a(t)$

$$\dot{\mathbf{b}}(t)=\sigma_b\mathbf{w}(t)$$

i.e., the derivative of the bias is white Gaussian noise (so-called random walk)

$$\mathbf{b}[k]=\mathbf{b}[k-1]+\sigma_{bd}\mathbf{w}[k]$$

$$\sigma_{bd}=\sigma_b\sqrt{\Delta t}$$

$$w[k]\sim N(0,1)$$

The biases are usually estimated with the other states

- can change every time the IMU is started
- can change due to temperature change, mechanical pressure, etc.

Trawny, Nikolas, and Stergios I. Roumeliotis. "Indirect Kalman filter for 3D attitude estimation.“
https://github.com/ethz-asl/kalibr/wiki/IMU-Noise-Model

IMU model: Integration

- Per component: {*t*} stands for {B}ody frame at time *t*

$$\mathbf{p}_{\mathrm{Wt}_2} = \boxed{\mathbf{p}_{\mathrm{Wt}_1}} + (t_2 - t_1)\boxed{\mathbf{v}_{\mathrm{Wt}_1}} + \iint_{t_1}^{t_2} \left(\boxed{\mathbf{R}_{\mathrm{Wt}}}(t)\left(\tilde{\mathbf{a}}(t) - \mathbf{b}^a(t)\right) + {}_{\mathrm{W}}\mathbf{g} \right) \mathrm{d}t^2$$

$$\mathbf{v}_{\mathrm{Wt}_2} = \boxed{\mathbf{v}_{\mathrm{Wt}_1}} + \int_{t_1}^{t_2} \left(\boxed{\mathbf{R}_{\mathrm{Wt}}}(t)\left(\mathbf{a}(t) - \mathbf{b}^a(t)\right) + {}_{\mathrm{W}}\mathbf{g} \right) \mathrm{d}t$$

- Depends on initial position and velocity
- The rotation R(t) is computed from the gyroscope

Rotation is more involved, will use quaternion as example:

$$\dot{\mathbf{q}}_{\mathrm{Wt}}(t) = \frac{1}{2}\Omega\left(\boldsymbol{\omega}(t)\right)\mathbf{q}_{\mathrm{Wt}}(t) \quad \Longrightarrow \quad \mathbf{q}_{\mathrm{Wt}_2}(t_2) = \Theta(t_1, t_2)\,\mathbf{q}_{\mathrm{Wt}_1}(t_1)$$

$\Theta(t_1, t_2)$ is the state transition matrix.

Trawny, Nikolas, and Stergios I. Roumeliotis. "Indirect Kalman filter for 3D attitude estimation."

IMU model: Integration

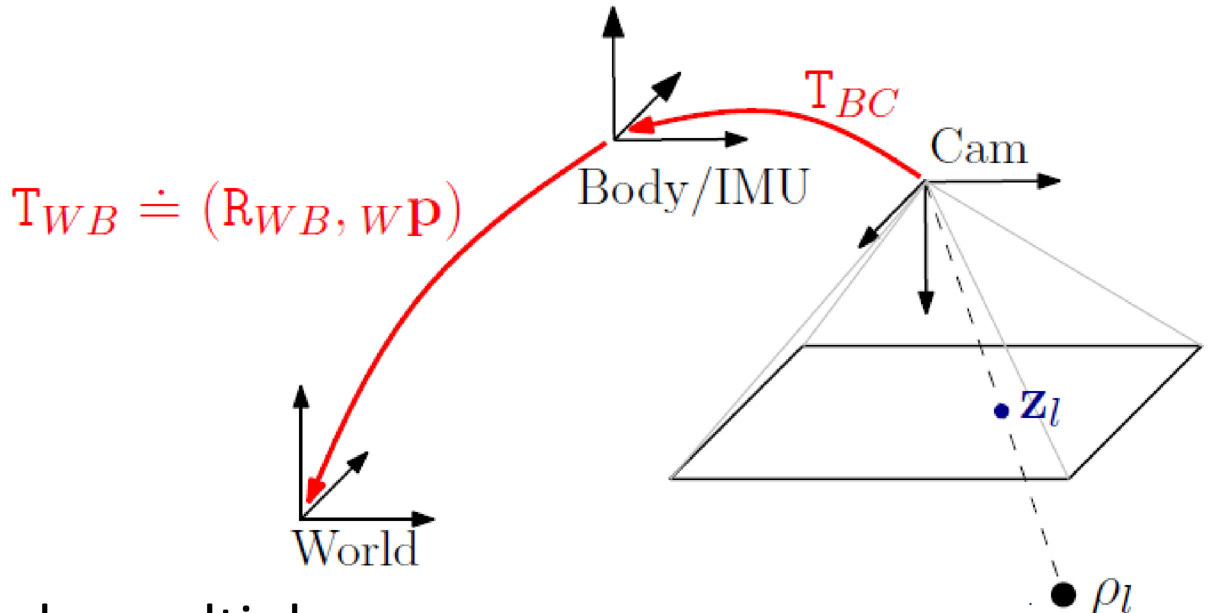
- Per component: {*t*} stands for {B}ody frame at time *t*

$$\mathbf{p}_{\mathrm{Wt}_2} = \boxed{\mathbf{p}_{\mathrm{Wt}_1}} + (t_2 - t_1)\boxed{\mathbf{v}_{\mathrm{Wt}_1}} + \int_{t_1}^{t_2}\int \left(\boxed{\mathbf{R}_{\mathrm{Wt}}}(t)\left(\tilde{\mathbf{a}}(t) - \mathbf{b}^a(t)\right) + {}_{\mathrm{w}}\mathbf{g}\right)\mathrm{d}t^2$$

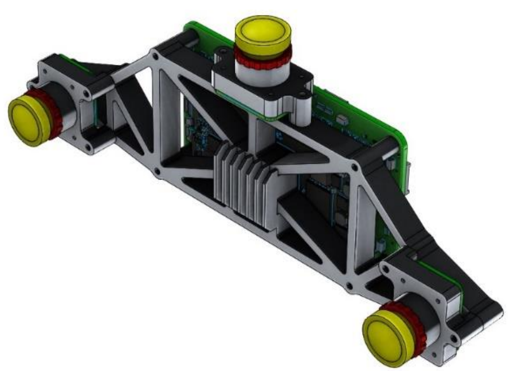
- Depends on initial position and velocity
- The rotation R(t) is computed from the gyroscope

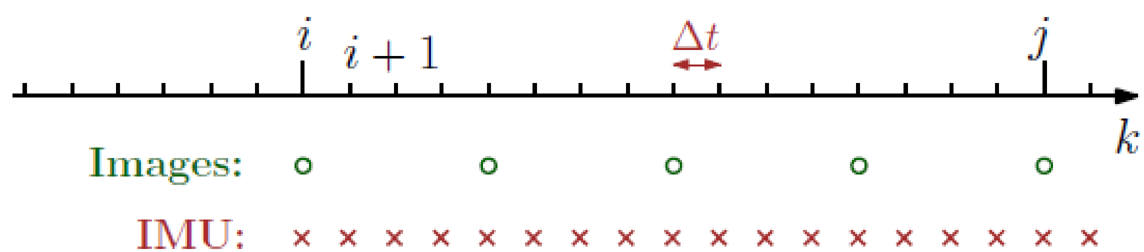
Trawny, Nikolas, and Stergios I. Roumeliotis. "Indirect Kalman filter for 3D attitude estimation."

Camera-IMU System



There can be multiple cameras.





Outline

- Introduction
- IMU model and Camera-IMU system
- Different paradigms
  - Closed-form solution
  - Filtering approaches
  - Maximum a posteriori estimation (non linear optimizers)
    - Fixed-lag Smoothing (aka sliding window estimators)
    - Full smoothing methods
- Camera-IMU extrinsic calibration and Synchronization

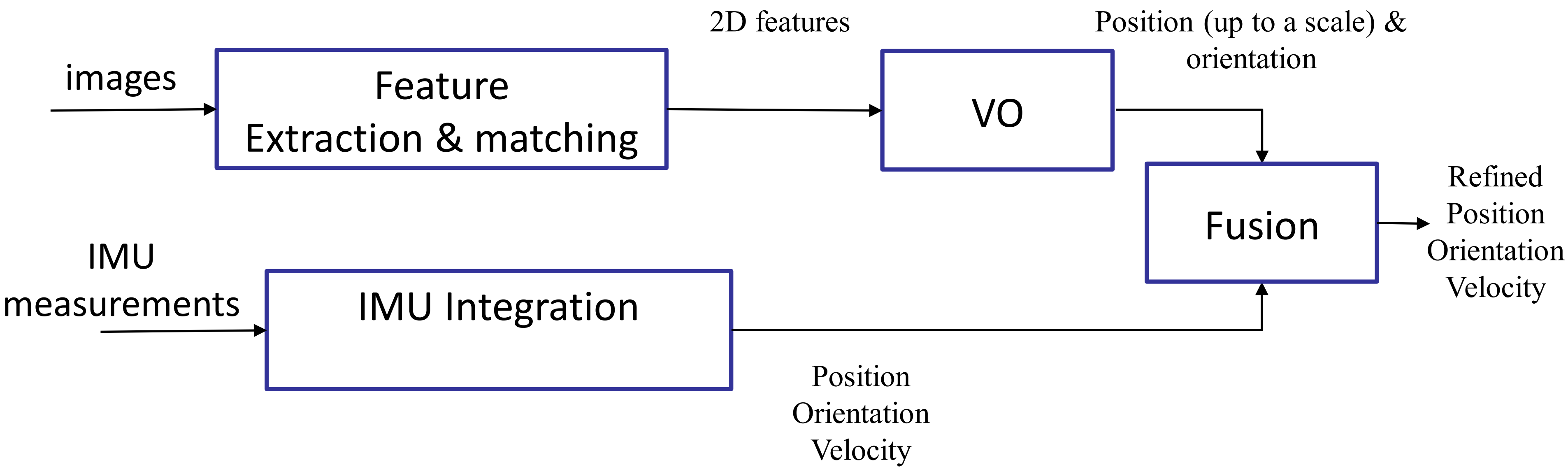
22

Different paradigms

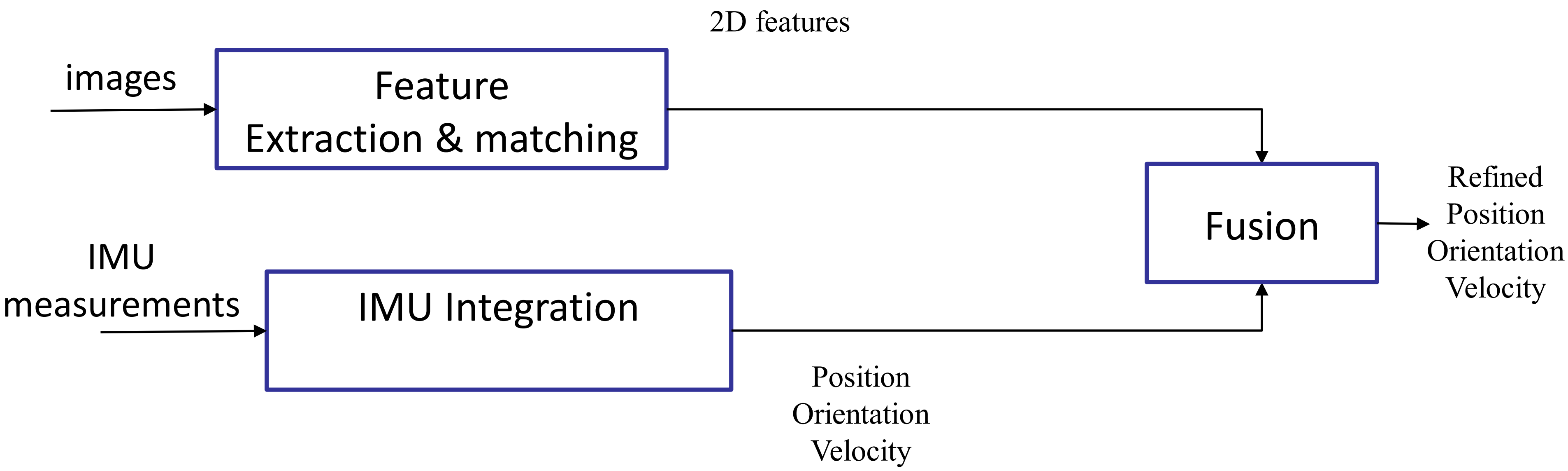
- **Loosely coupled**:
  - Treats VO and IMU as two separate (not coupled) black boxes
    - Each black box estimates pose and velocity from visual (up to a scale) and inertial data (absolute scale)
- **Tightly coupled**:
  - Makes use of the raw sensors' measurements:
    - 2D features
    - IMU readings
    - More accurate
    - More implementation effort
  - In the following slides, we will only see **tightly coupled approaches**

23

The Loosely Coupled Approach



The Tightly Coupled Approach



25

Filtering: Visual Inertial Formulation

System states:

Tightly coupled: $\mathbf{X} = \left[{}_{\mathrm{W}}\mathbf{p}(t);\ \mathbf{q}_{\mathrm{WB}}(t);\ {}_{\mathrm{W}}\mathbf{v}(t); \mathbf{b}^{a}(t); \mathbf{b}^{g}(t);\ \boxed{{}_{\mathrm{W}}\mathbf{L}_1;\ {}_{\mathrm{W}}\mathbf{L}_2; \ldots ;{}_{\mathrm{W}}\ \mathbf{L}_K} \right]$

Loosely coupled: $\mathbf{X} = \left[{}_{\mathrm{W}}\mathbf{p}(t);\ \mathbf{q}_{\mathrm{WB}}(t);\ {}_{\mathrm{W}}\mathbf{v}(t); \mathbf{b}^{a}(t); \mathbf{b}^{g}(t) \right]$

Corke, An Introduction to Inertial and Visual Sensing, IJRR’07

26

Outline

- Introduction
- IMU model and Camera-IMU system
- Different paradigms
  - Closed-form solution
  - Filtering approaches
  - Maximum a posteriori estimation (non linear optimizers)
    - Fixed-lag Smoothing (aka sliding window estimators)
    - Full smoothing methods
- Camera-IMU extrinsic calibration and Synchronization

Closed-form Solution (1D case)

- The absolute pose x is known up to a scale s, thus

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

- From the IMU

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

- By equating them

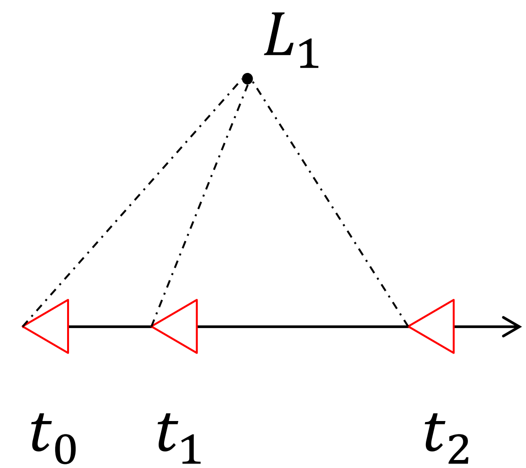
$$s\tilde{x} \;= x_0 + v_0(t_1 - t_0) + \iint_{t_0}^{t_1} a(t)dt$$

- As shown in [Martinelli'14], for 6DOF, both s and v_0 can be determined in closed form from a **single feature observation and 3 views**. x_0 can be set to 0.

Martinelli, Closed-form solution of visual-inertial structure from motion, International Journal of Computer Vision, 2014

Closed-form Solution (1D case)

$$\begin{cases} s\widetilde{x_1} = v_0(t_1 - t_0) + \iint_{t_0}^{t_1} a(t)dt \\ s\widetilde{x_2} = v_0(t_2 - t_0) + \iint_{t_0}^{t_2} a(t)dt \end{cases}$$



$$\begin{bmatrix} \widetilde{x_1} & (t_0 - t_1) \\ \widetilde{x_2} & (t_0 - t_2) \end{bmatrix} \begin{bmatrix} s \\ v_0 \end{bmatrix} = \begin{bmatrix} \iint_{t_0}^{t_1} a(t)dt \\ \iint_{t_0}^{2} a(t)dt \end{bmatrix}$$

Martinelli, Closed-form solution of visual-inertial structure from motion, International Journal of Computer Vision, 2014

Closed-form Solution (general case)

- Considers N feature observations and 6DOF case
- Can be used to initialize filters and smoothers (which always need an initialization point)
- 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:

- 3D Point distances (wrt the first camera)
- Absolute scale,
- Initial velocity,
- Gravity vector,
- Biases

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

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

- Martinelli, Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination, TRO'12
- Martinelli, Closed-form solution of visual-inertial structure from motion, Int. Journal of Comp. Vision, JCV'14
- Kaiser, Martinelli, Fontana, Scaramuzza, Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation, IEEE RAL'17

30

Outline

- Introduction
- IMU model and Camera-IMU system
- Different paradigms
  - Closed-form solution
  - Filtering approaches
  - Maximum a posteriori estimation (non linear optimizers)
    - Fixed-lag Smoothing (aka sliding window estimators)
    - Full smoothing methods
- Camera-IMU extrinsic calibration and Synchronization

31

Different paradigms

Filtering	**Fixed-lag Smoothing**	**Full smoothing**
Only updates the most recent states • (e.g., extended Kalman filter)	Optimizes window of states • Marginalization • Nonlinear least squares optimization	Optimize all states • Nonlinear Least squares optimization
×1 Linearization ×Accumulation of linearization errors ×Gaussian approximation of marginalized states ✓Fastest	✓Re-Linearize ×Accumulation of linearization errors ×Gaussian approximation of marginalized states ✓Fast	✓Re-Linearize ✓Sparse Matrices ✓Highest Accuracy ×Slow (but fast with GTSAM)

Filtering: Kalman Filter in a Nutshell

- Assumptions: **linear system, Gaussian noise**

System dynamics

$$x(k) = A(k-1)x(k) + u(k-1) + v(k-1)$$

$$z(k) = H(k)x(k) + w(k)$$

x(*k*): state

u(*k*): control input, can be 0

z(k): measurement

$$x(0) \sim N(x_0, P_0)$$

$$v(k) \sim N(0, Q(k))$$

$$w(k) \sim N(0, R(k))$$

Kalman Filter

$$x_m(0) = x_0, P_m(0) = P_0$$

Prediction

$$\hat{x}_p(k) = A(k-1)\hat{x}_m(k-1) + u(k-1)$$

$$P_p(k) = A(k-1)P_m(k-1)A^T(k-1) + Q(k-1)$$

Measurement update

$$P_m(k) = \left(P_p(k) + H^T(k)R^{-1}(k)H(k)\right)^{-1}$$

$$\hat{x}_m(k) = \hat{x}_p(k) + \boxed{P_m(k)H^T(k)R^{-1}(k)}\left(z(k) - H(k)\hat{x}_p(k)\right)$$

Weight between the model prediction and measurement

Filtering: Kalman Filter in a Nutshell

➢ Nonlinear system: linearization

System dynamics

$$x(k) = q_{k-1}\left(x(k-1), u(k-1), v(k-1)\right)$$
$$z(k) = h_k\left(x(k), w(k)\right)$$

Process and measurement noise and initial state are Gaussian.

Key idea:

- Linearize around the estimated states
- *A(k) L(k) H(k) M(k)* are partial derivatives with respect to states and noise

Extended Kalman Filter

Prediction

$$\hat{x}_p(k) = q_{k-1}\left(\hat{x}_m(k-1), u(k-1), 0\right)$$
$$P_p(k) = A(k-1)P_m(k-1)A^T(k-1) + L(k-1)Q(k-1)L^T(k-1)$$

Measurement update

$$K(k) = P_p(k)H^T(k)(H(k)P_p(k)H^T(k) + M(k)R(k)M^T(k))^{-1}$$
$$\hat{x}_m(k) = \hat{x}_p(k) + K(k)\left(z(k) - h_k\left(\hat{x}_p, 0\right)\right)$$
$$P_m(k) = \left(I - K(k)H(k)\right)P_p(k)$$

Filtering: Visual Inertial Formulation

System states:

Tightly coupled: $\mathbf{X} = \left[\boxed{{}_{\mathrm{W}}\mathbf{p}(t);\ \mathbf{q}_{\mathrm{WB}}(t);\ {}_{\mathrm{W}}\mathbf{v}(t); \mathbf{b}^{a}(t); \mathbf{b}^{g}(t);}\ {}_{\mathrm{W}}\mathbf{L}_1;\ {}_{\mathrm{W}}\mathbf{L}_2; \ldots; {}_{\mathrm{W}}\mathbf{L}_K \right]$

Loosely coupled: $\mathbf{X} = \left[{}_{\mathrm{W}}\mathbf{p}(t);\ \mathbf{q}_{\mathrm{WB}}(t);\ {}_{\mathrm{W}}\mathbf{v}(t); \mathbf{b}^{a}(t); \mathbf{b}^{g}(t) \right]$

Process Model: from IMU

- Integration of IMU states (rotation, position, velocity)
- Propagation of IMU noise
  - needed for calculating the Kalman Filter gain

Filtering: Visual Inertial Formulation

Measurement Model: from camera

Transform point to camera frame

$$\begin{bmatrix} {}_{C}x \\ {}_{C}y \\ {}_{C}z \end{bmatrix} = \mathbf{R}_{\mathrm{CB}}\Big(\underbrace{\mathbf{R}_{\mathrm{BW}}\left({}_{\mathrm{W}}\mathbf{L} - {}_{\mathrm{W}}\mathbf{p}\right)}_{{}_{\mathrm{B}}\mathbf{L}} - \mathbf{p}_{\mathrm{CB}}\Big)$$

$$\mathbf{H}_{\mathrm{Landmark}} = \mathbf{R}_{\mathrm{CB}}\mathbf{R}_{\mathrm{BW}} = \mathbf{R}_{\mathrm{CW}}$$

$$\mathbf{H}_{\mathrm{pose}} = \mathbf{R}_{\mathrm{CB}}\begin{bmatrix} -\mathbf{R}_{\mathrm{BW}} & \left[{}_{\mathrm{B}}\mathbf{L}\right]_{\times} \end{bmatrix}$$

Pinhole projection (without distortion)

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \dfrac{{}_{C}x}{{}_{C}z} + c_x \\ f_y \dfrac{{}_{C}y}{{}_{C}z} + c_y \end{bmatrix}$$

$$\mathbf{H}_{\mathrm{proj}} = \begin{bmatrix} f_x \dfrac{1}{z} & 0 & -f_x \dfrac{x}{z^2} \\ 0 & f_y \dfrac{1}{z} & -f_y \dfrac{y}{z^2} \end{bmatrix}$$

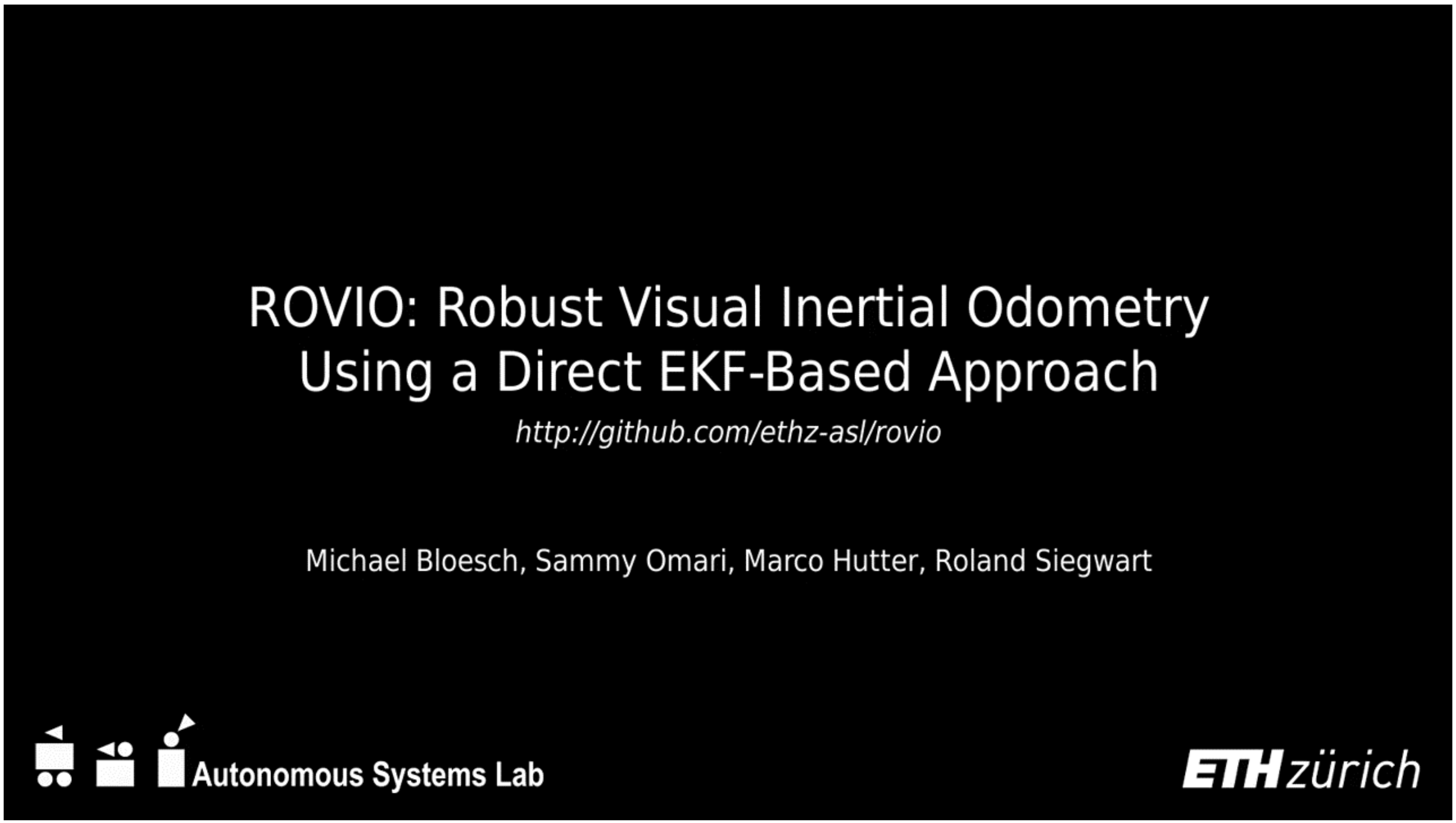
Drop C for clarity

$$\mathbf{H}_{\mathbf{X}} = \mathbf{H}_{\mathrm{proj}}\mathbf{H}_{\mathrm{pose}}$$

$$\mathbf{H}_{\mathbf{L}} = \mathbf{H}_{\mathrm{proj}}\mathbf{H}_{\mathrm{Landmark}}$$

Filtering: ROVIO

- EKF state: $\mathbf{X} = \left[{}_{\mathrm{W}}\mathbf{p}(t);\ \mathbf{q}_{\mathrm{WB}}(t);\ {}_{\mathrm{W}}\mathbf{v}(t); \mathbf{b}^{a}(t); \mathbf{b}^{g}(t);\ {}_{\mathrm{W}}\mathbf{L}_1;\ {}_{\mathrm{W}}\mathbf{L}_2; \ldots; {}_{\mathrm{W}}\mathbf{L}_K \right]$
- Minimizes the photometric error instead of the reprojection error



Bloesch, Michael, et al. "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback", IJRR'17

Filtering: Potential Problems

- Wrong linearization point
  - Linearization depends on the current estimates of states, which may be erroneous
  - Linearization around different values of the same variable leads to estimator inconsistency (wrong observability/covariance estimation)
- Wrong covariance/initial states
  - Intuitively, wrong weights for measurements and prediction
  - May be overconfident/underconfident
- Explosion of number of states
  - Roughly cubic in the number of the states
  - Each 3D point: 3 variables
  - → a **few landmarks** (~20) are typically tracked to allow real-time operation

Filtering: Problems

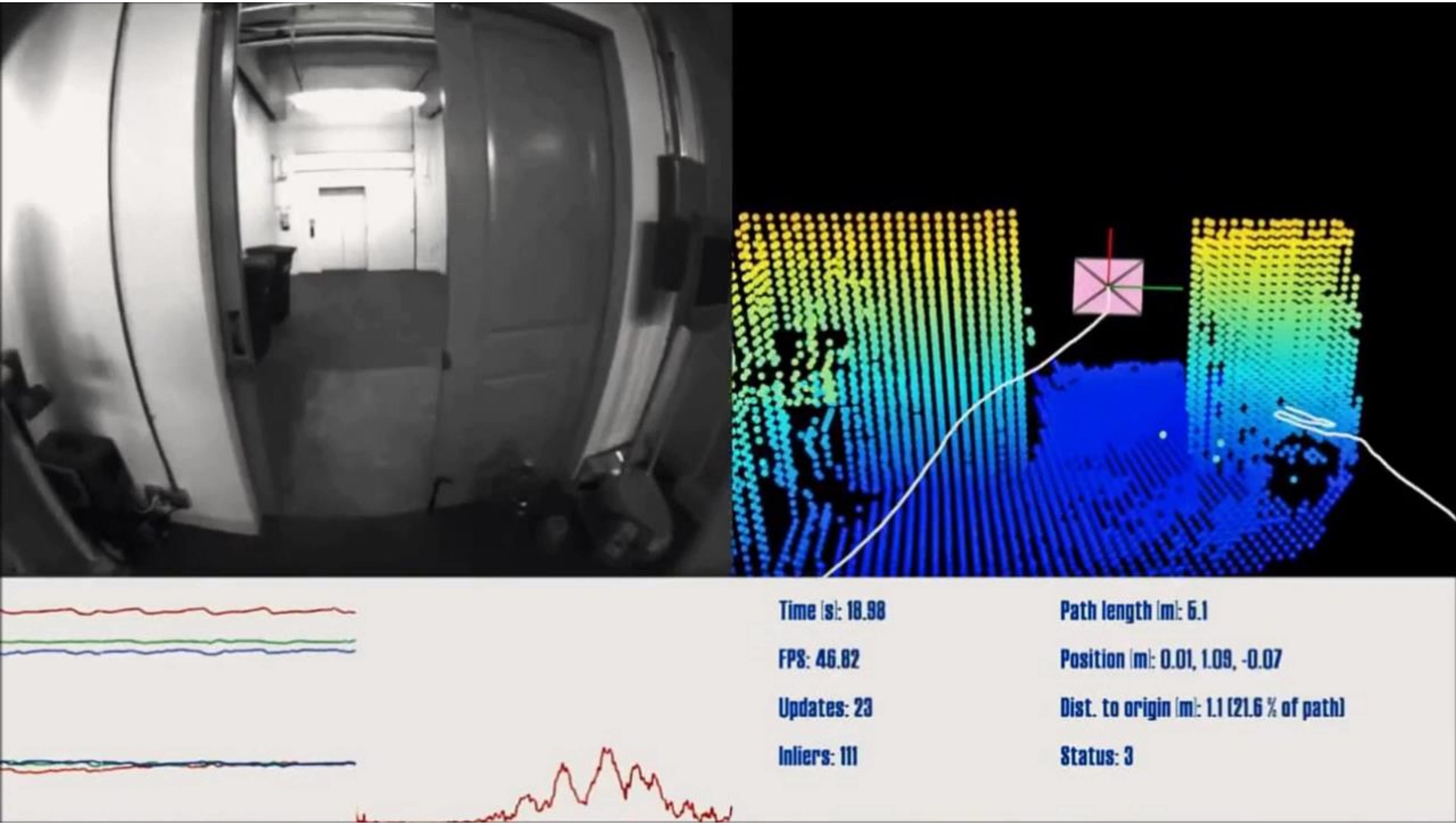
- **Alternative: MSCKF** [Mourikis & Roumeliotis, ICRA'07]: used in Google Tango
  - Keeps a window of recent states and updates them using EKF
  - incorporate visual observations without including point positions into the states

Mourikis & Roumeliotis, A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation, TRO'16

Li, Mingyang, and Anastasios I. Mourikis, High-precision, consistent EKF-based visual–inertial odometry, IJRR'13

Filtering: Google Tango

Project Tango



Mourikis & Roumeliotis, A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation, TRO'16

Li, Mingyang, and Anastasios I. Mourikis, High-precision, consistent EKF-based visual–inertial odometry, IJRR'13

Outline

- Introduction
- IMU model and Camera-IMU system
- Different paradigms
  - Closed-form solution
  - Filtering approaches
  - Maximum a posteriori estimation (non linear optimizers)
    - Fixed-lag Smoothing (aka sliding window estimators)
    - Full smoothing methods
- Camera-IMU extrinsic calibration and Synchronization

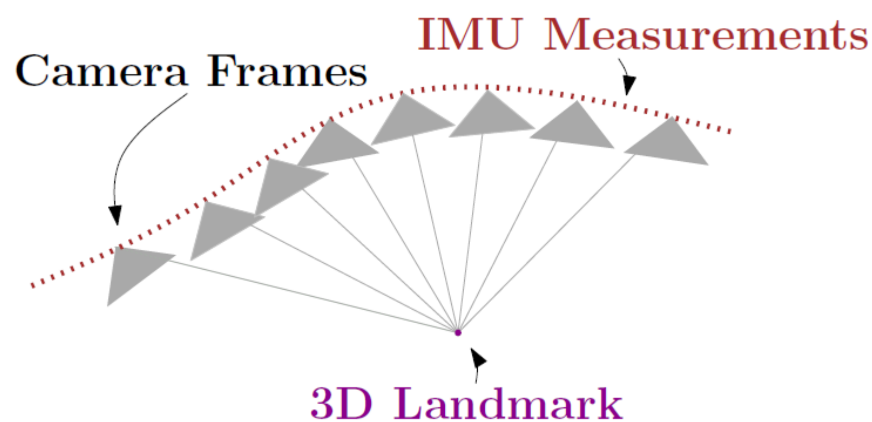
41

Maximum A Posteriori (MAP) Estimation

- Fusion solved as a *non-linear optimization problem*
- Increased accuracy over filtering methods

$$x_k = f(x_{k-1})$$
$$z_i = h(x_{i_k}, l_{i_j})$$

$X = \{x_1, \ldots x_N\}$: robot states
$L = \{l_1, \ldots\}$: 3D points
$Z = \{z_i, \ldots z_M\}$: features & IMU measurements



$$\{X^*, L^*\} = \underset{\{X,L\}}{\operatorname{argmax}} P(X, L \mid Z)$$
$$= \underset{\{X,L\}}{\arg\min} \left\{ \underbrace{\sum_{k=1}^{N} \left\| f(x_{k-1}) - x_k \right\|_{\Lambda_k}^2}_{\textit{IMU residuals}} + \underbrace{\sum_{i=1}^{M} \left\| h(x_{i_k}, l_{i_j}) - z_i \right\|_{\Sigma_i}^2}_{\textit{Reprojection residuals}} \right\}$$

[Jung, CVPR'01] [Sterlow'04] [Bryson, ICRA'09] [Indelman, RAS'13] [Patron-Perez, IJCV'15][Leutenegger, RSS'13-IJRR'15] [Forster, RSS'15, TRO'16]

MAP: a nonlinear least squares problem

- Bayesian Theorem

$$\boxed{P(A|B)} = \frac{P(B|A)P(A)}{P(B)}$$

Applied to state estimation problem:

- X: states (position, attitude, velocity, and 3D point position)
- Z: measurements (feature positions, IMU readings)

Max a Posteriori: given the observation, what is the optimal estimation of the states?

- Gaussian Property: for **iid** variables

$$f(x_1,\ldots,x_k \mid \mu,\sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{\Sigma(x_i-\mu)^2}{2\sigma^2}}$$

Maximizing the probability is equivalent to minimizing the square root sum

MAP: a nonlinear least squares problem

- SLAM as a MAP problem

$$x_k = f(x_{k-1})$$
$$z_i = h(x_{i_k}, l_{i_j})$$

X = {x_1,…x_N}: robot states
L = {l_1,…}: 3D points
Z = {z_i, … z_M}: feature positions

$$\begin{aligned}
P(X, L \mid Z) &\propto P(Z \mid X, L) P(X, L) \\
&\propto \left(\prod_{i=1}^{M} P(z_i \mid X, L) \right) P(X) \\
&\propto P(x_0) \left(\prod_{i=1}^{M} P(z_i \mid x_{i_k}, l_{i_j}) \right) \left(\prod_{k=2}^{N} P(x_k \mid x_{k-1}) \right)
\end{aligned}$$

- *X L* are independent, and no prior information about *L*
- Measurements are independent
- Markov process model

MAP: a nonlinear least squares problem

- SLAM as a least squares problem

$$P(X,L|Z) \propto P(x_0)\left(\prod_{i=1}^{M} P(z_i \mid x_{i_k}, l_{i_j})\right)\left(\prod_{k=2}^{N} P(x_k \mid x_{k-1})\right)$$

Without the prior, applying the property of Gaussian distribution:

$$\{X^*, L^*\} = \underset{\{X,L\}}{\operatorname{argmax}} P(X,L|Z)$$

$$= \underset{\{X,L\}}{\arg\min} \left\{ \sum_{k=1}^{N} \left\| f(x_{k-1}) - x_k \right\|_{\Lambda_k}^{2} + \sum_{i=1}^{M} \left\| h(x_{i_k}, l_{i_j}) - z_i \right\|_{\Sigma_i}^{2} \right\}$$

- Notes:
  - Normalize the residuals with the variance of process noise and measurement noise (so-called **Mahalanobis distance**)

MAP: Nonlinear optimization

- Gauss-Newton method

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{M} \left\| f_i(\boldsymbol{\theta}) - \mathbf{z}_i \right\|^2$$

Solve it iteratively

$$\boldsymbol{\varepsilon}^* = \arg\min_{\varepsilon} \sum_{i=1}^{M} \left\| f_i(\boldsymbol{\theta}^s + \boldsymbol{\varepsilon}) - \mathbf{z}_i \right\|^2$$

$$\boldsymbol{\theta}^{s+1} = \boldsymbol{\theta}^s + \boldsymbol{\varepsilon}$$

Applying first-order approximation:

$$\boldsymbol{\varepsilon}^* = \arg\min_{\varepsilon} \sum_{i=1}^{M} \left\| f_i(\boldsymbol{\theta}^s) - \mathbf{z}_i + \mathbf{J}_i \boldsymbol{\varepsilon} \right\|^2$$

$$= \arg\min_{\varepsilon} \sum_{i=1}^{M} \left\| \mathbf{r}_i(\boldsymbol{\theta}^s) + \mathbf{J}_i \boldsymbol{\varepsilon} \right\|^2$$

$$\boldsymbol{\varepsilon}^* = -(\boxed{\mathbf{J}^T \mathbf{J}})^{-1} \mathbf{J}^{\mathrm{T}} \mathbf{r}(\boldsymbol{\theta})$$

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_1 \\ \mathbf{J}_2 \\ \cdots \\ \mathbf{J}_M \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \cdots \\ \mathbf{r}_M \end{bmatrix}$$

MAP: visual inertial formulation

- States

$$\mathbf{X}_R\left(k\right)=\left[\mathbf{p}_{\mathrm{WB}}\left[k\right],\mathbf{q}_{\mathrm{WB}}\left[k\right],\mathbf{v}_{\mathrm{WB}}[\mathrm{k}],\mathbf{b}^a\left[k\right],\mathbf{b}^g\left[k\right]\right]$$

$$\mathbf{X}_{\mathrm{L}}=\left[\mathbf{L}_{\mathrm{W1}},\mathbf{L}_{\mathrm{W2}},\ldots,\mathbf{L}_{\mathrm{WL}}\right]$$

Combined: $\mathbf{X}=\left[\mathbf{X}_{\mathrm{R}}\left[1\right],\mathbf{X}_{\mathrm{R}}\left[2\right],\ldots,\mathbf{X}_{\mathrm{R}}\left[k\right],\mathbf{X}_{\mathrm{L}}\right]$

- Dynamics Jacobians
  - IMU integration w.r.t $\mathbf{x}_{k-1}$
  - Residual w.r.t. $\mathbf{x}_k$

$$\mathbf{f}\left(\mathbf{x}_{k-1}\right)-\mathbf{x}_K$$

- Measurements Jacobians (same as filtering method)
  - Feature position w.r.t. pose
  - Feature position w.r.t. 3D coordinates

$$\mathbf{h}\left(\mathbf{x}_{ik},\mathbf{L}_{ij}\right)-\mathbf{z}_i$$

Fixed-lag smoothing: Basic Idea

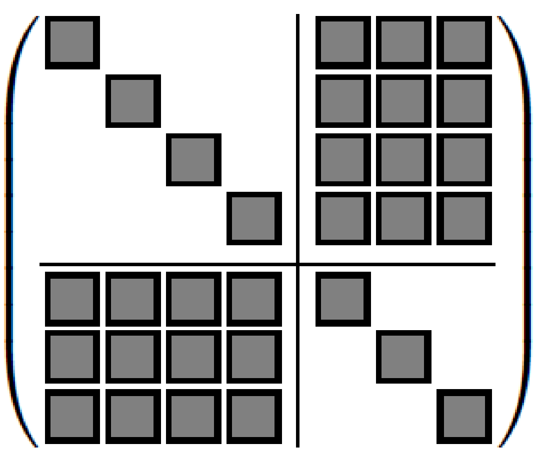
- Recall MAP estimation

$$\boldsymbol{\varepsilon}^* = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^{\mathrm{T}}\mathbf{r}(\boldsymbol{\theta})$$

$\mathbf{J}^T\mathbf{J}$ is also called the Hessian matrix.

- Hessian for full bundle adjustment: n x n, n number of all the states

pose, velocity | landmarks



If only part of the states are of interest, can we think of a way for simplification?

Fixed-lag smoothing: Marginalization

- Schur complement

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

$\bar{D} = D - CA^{-1}B$ Schur complement of A in M

$\bar{A} = A - BD^{-1}C$ Schur complement of D in M

- Reduced linear system

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ -CA^{-1} & 1 \end{pmatrix}\begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -CA^{-1} & 1 \end{pmatrix}\begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix}$$

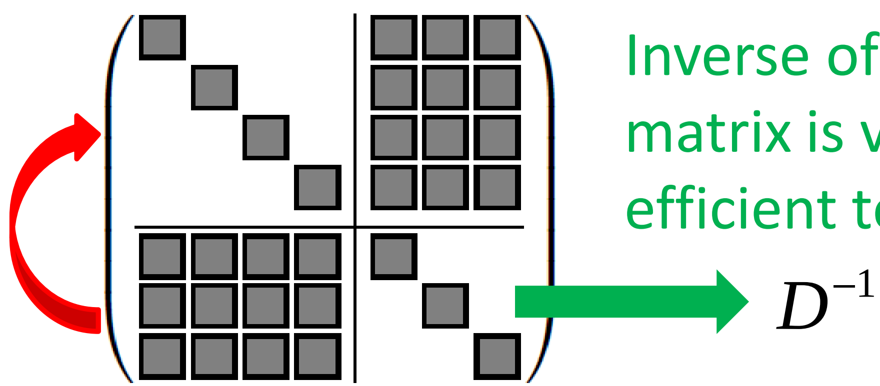
$$\begin{pmatrix} A & B \\ 0 & \bar{D} \end{pmatrix} = \begin{pmatrix} b_1 \\ \bar{b}_2 \end{pmatrix} \qquad \bar{b}_2 = b_2 - CA^{-1}b_1$$

We can then just solve for x_2, and (optionally) solve for x_1 by back substitution.

Fixed-lag smoothing: Marginalization

- Generalized Schur complement
  - Any principal submatrix: selecting n rows and n columns of the same index (i.e., select any states to marginalize)
  - Nonsingular submatrix: use generalized inverse (e.g., Moore–Penrose pseudoinverse)
- Special structure of SLAM

Marginalization causes fill-in, no longer maintaining the sparse structure.



Inverse of diagonal matrix is very efficient to calculate.

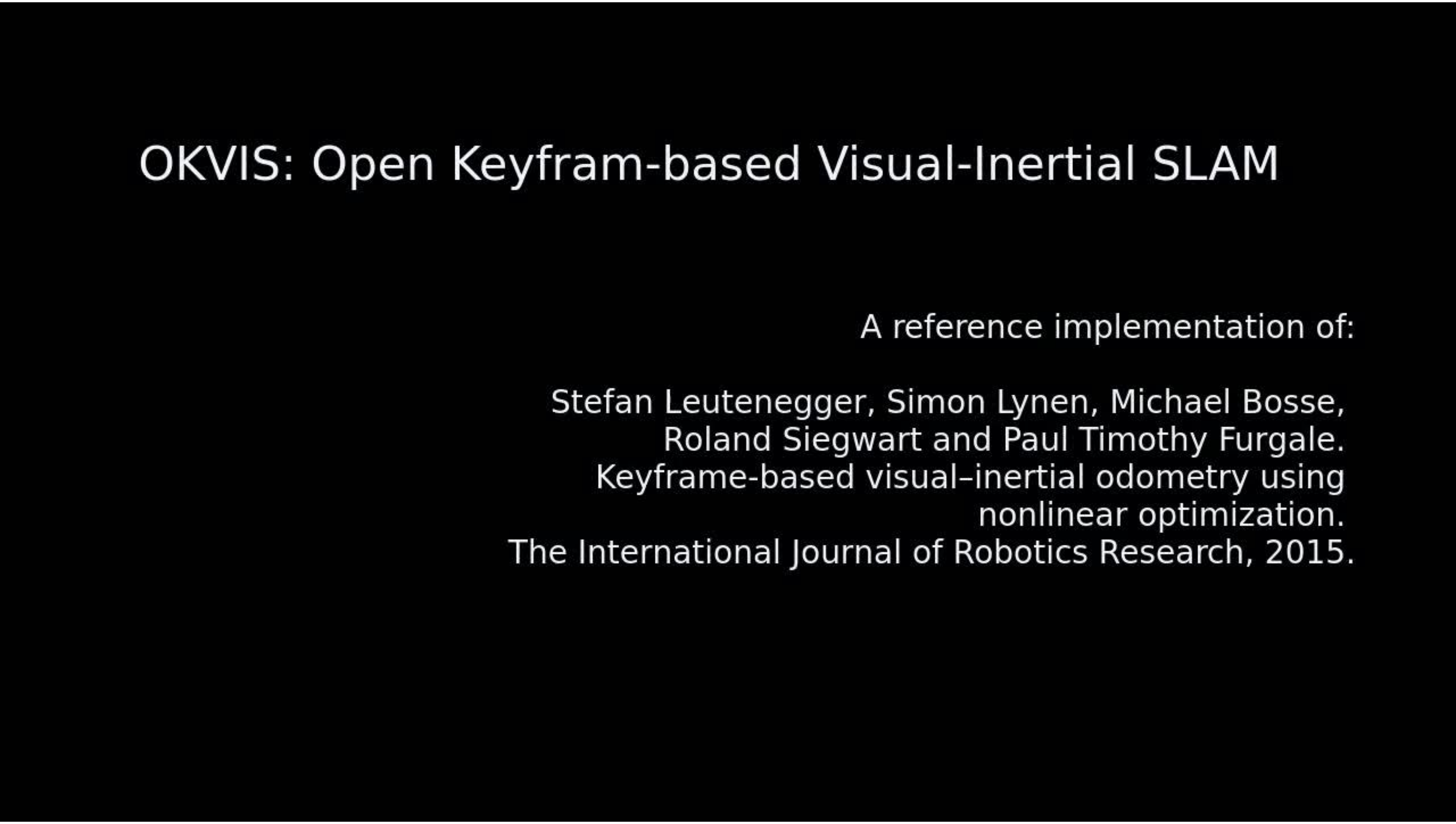
Fixed-lag smoothing: Implementation

- States and formulations are similar to MAP estimation.
- Which states to marginalize?
  - Old states: keep a window of recent frames
  - Landmarks: structureless
- Marginalizing states vs. dropping the states
  - Dropping the states: loss of information, not optimal
  - Marginalization: optimal if there is no linearization error, but introduces fill-in, causing performance penalty

Therefore, dropping states is also used to **trade accuracy for speed**.

Leutenegger, Stefan, et al. "Keyframe-based visual–inertial odometry using nonlinear optimization."

Fixed-lag smoothing: OKVIS



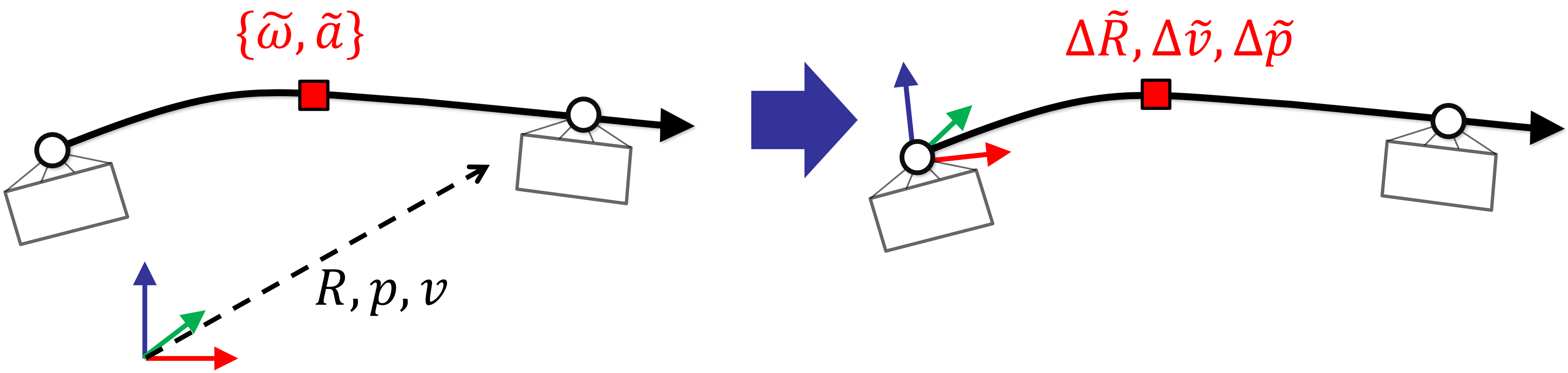
Leutenegger, OKVIS: Open Keyframe-based Visual-Inertial SLAM, IJRR'15

MAP: why it is slow

- Re-linearization
  - Need to recalculate the Jacobian for each iteration
  - But it is also an important reason why MAP is accurate
- The number of states is large
  - Will see next: fix-lag smoothing and marginalization
- Re-integration of IMU measurements
  - The integration from *k* to *k+1* is related to the state estimation at time *k*
  - Preintegration

    Lupton, Todd, and Salah Sukkarieh. "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions."
    Forster, Christian, et al. "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation."

MAP: IMU Preintegration



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

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

$$e_v = \hat{v}(\tilde{\omega}, \tilde{a}, v_{k-1}) - v_k$$

$$e_p = \underbrace{\hat{p}(\tilde{\omega}, \tilde{a}, p_{k-1})}_{\textbf{Predicted}} - \underbrace{p_k}_{\textbf{Estimate}}$$

Repeat integration when previous state changes!

Preintegration:
Evaluate **relative errors**:

$$e_R = \Delta\tilde{R}^T \, \Delta R$$

$$e_v = \Delta\tilde{v} - \Delta v$$

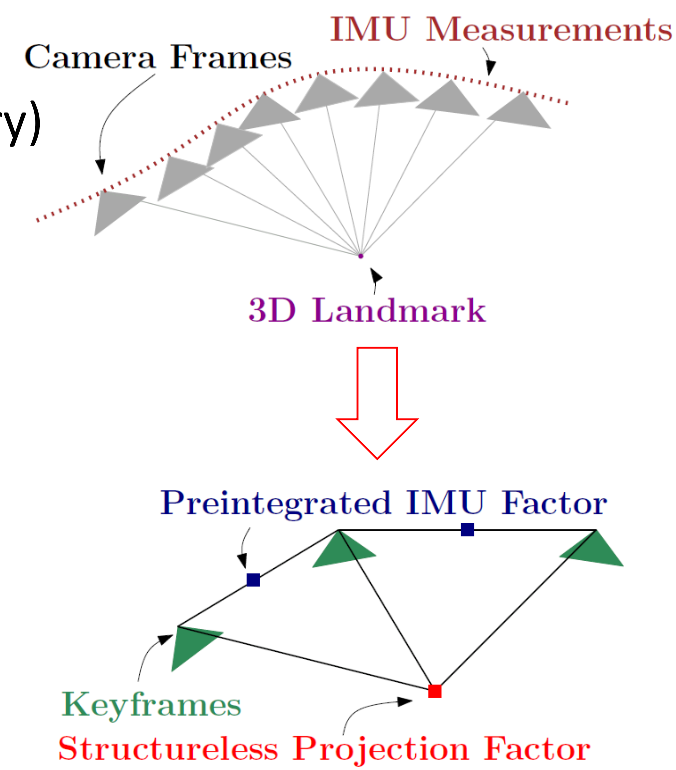
$$e_p = \Delta\tilde{p} - \Delta p$$

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

Full Smoothing: SVO+IMU Preintegration

Solves the same optimization problem but:

- Keeps all the frames (from the start of the trajectory)
- To make the optimization efficient
  - it makes the graph sparser using keyframes
  - pre-integrates the IMU data between keyframes
- Optimization salved using factor graphs (**GTSAM**)
  - Very fast because it only optimizes the poses which are affected by a new observation



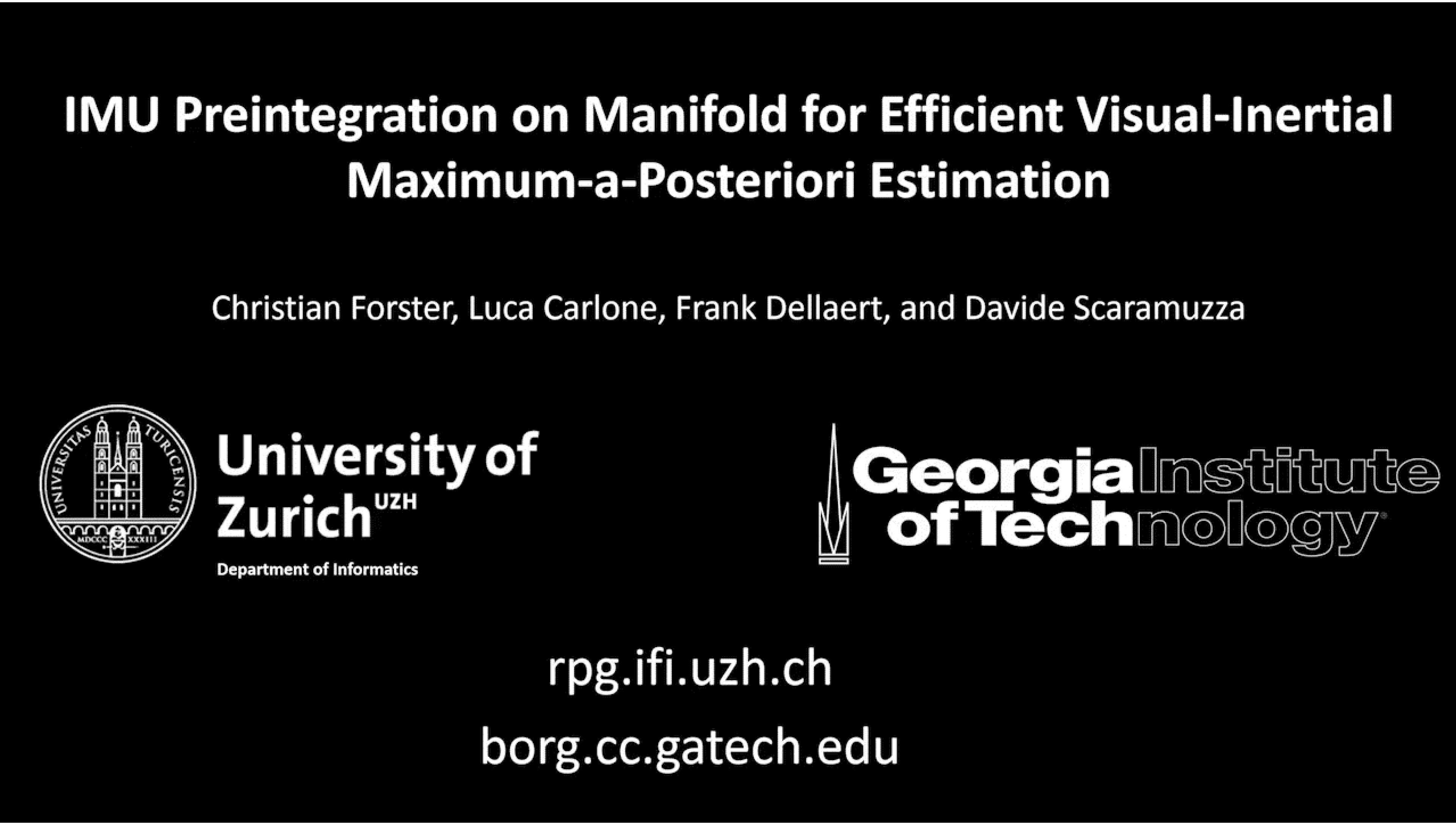
$$\{X^*, L^*\} = \underset{\{X,L\}}{\operatorname{argmax}}\, P(X, L \mid Z)$$

$$= \underset{\{X,L\}}{\arg\min} \left\{ \sum_{k=1}^{N} \left\| f(x_{k-1}) - x_k \right\|_{\Lambda_k}^2 + \sum_{i=1}^{M} \left\| h(x_{i_k}, l_{i_j}) - z_i \right\|_{\Sigma_i}^2 \right\}$$

IMU residuals *Reprojection residuals*

Forster, Carlone, Dellaert, Scaramuzza, On-Manifold Preintegration for Real-Time Visual-Inertial Odometry, IEEE Transactions on Robotics, Feb. 2017.

SVO + IMU Preintegration

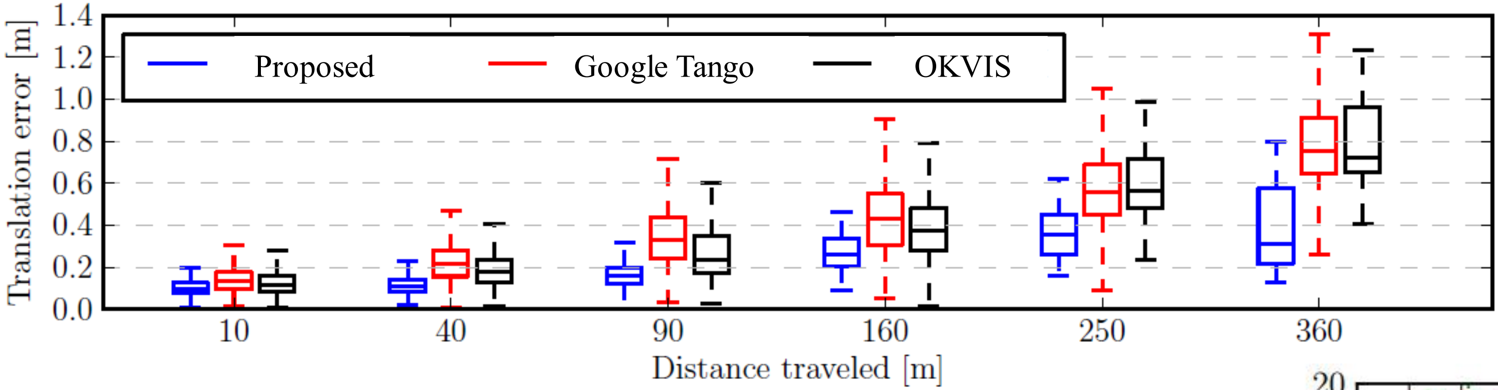


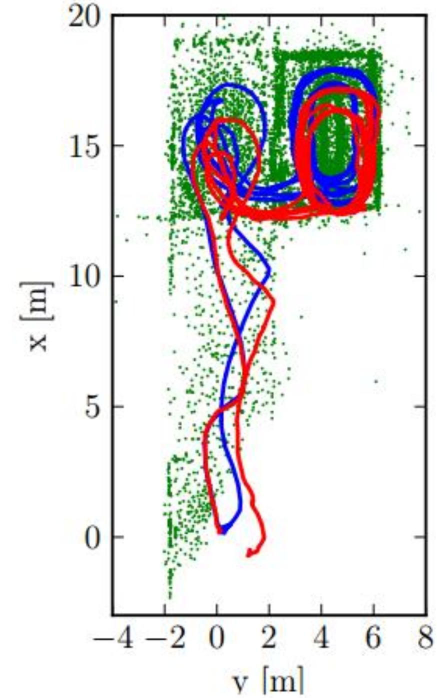
Forster, Carlone, Dellaert, Scaramuzza, On-Manifold Preintegration for Real-Time Visual-Inertial Odometry, IEEE Transactions on Robotics, Feb. 2017.

56

SVO + IMU Preintegration

Accuracy: 0.1% of the travel distance





Forster, Carlone, Dellaert, Scaramuzza, On-Manifold Preintegration for Real-Time Visual-Inertial Odometry, IEEE Transactions on Robotics, Feb. 2017.

57

Visual-Inertial Fusion: further reading and code

- **Closed form solution**:
  - for 6DOF motion both s and v_0 can be determined **1 feature observation and at least 3 views** [Martinelli, TRO'12, IJCV'14, RAL'16]
  - Can be used to **initialize filters and smoothers**
- **Filters:** *update only last state → fast if number of features is low (~20)*
  - [Mourikis, ICRA'07, CVPR'08], [Jones, IJRR'11] [Kottas, ISER'12][Bloesch, IROS'15] [Wu et al., RSS'15], [Hesch, IJRR'14], [Weiss, JFR'13]
  - **Open source**: **ROVIO** [Bloesch, IROS'15, IJRR'17], **MSCKF** [Mourikis, ICRA'07] (i.e., Google Tango)
- **Fixed-lag smoothers:** *update a window of states → slower but more accurate*
  - [Mourikis, CVPR'08] [Sibley, IJRR'10], [Dong, ICRA'11], [Leutenegger, RSS'13-IJRR'15]
  - **Open source: OKVIS** [Leutenegger, RSS'13-IJRR'15]
- **Full-smoothing methods:** *update entire history of states → slower but more accurate*
  - [Jung, CVPR'01] [Sterlow'04] [Bryson, ICRA'09] [Indelman, RAS'13] [Patron-Perez, IJCV'15] [Forster, RSS'15, TRO'16]
  - **Open source: SVO+IMU** [Forster, TRO'17]

Open Problem: consistency

- Filters
  - Linearization around different values of the same variable may lead to error
- Smoothing methods
  - May get stuck in local minima

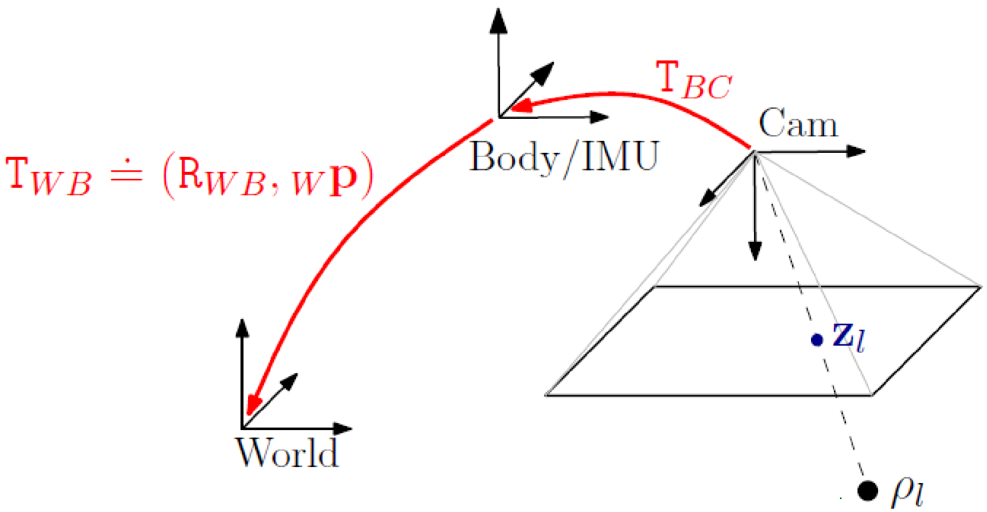
59

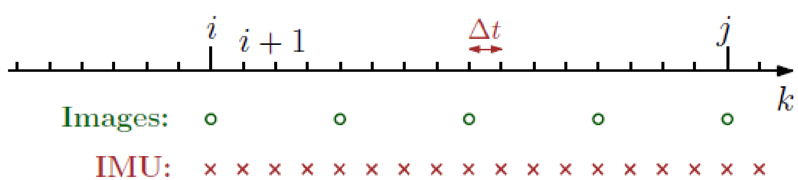
Outline

- Introduction
- IMU model and Camera-IMU system
- Different paradigms
  - Closed-form solution
  - Filtering approaches
  - Maximum a posteriori estimation (non linear optimizers)
    - Fixed-lag Smoothing (aka sliding window estimators)
    - Full smoothing methods
- Camera-IMU extrinsic calibration and Synchronization

Camera-IMU calibration

- **Goal**: estimate the rigid-body transformation $\boldsymbol{T_{BC}}$ and delay t_d between a camera and an IMU rigidly attached. Assume that the camera has already been intrinsically calibrated.
- **Data**:
  - Image points of detected calibration pattern (checkerboard).
  - IMU measurements: accelerometer $\{a_k\}$ and gyroscope $\{\omega_k\}$.

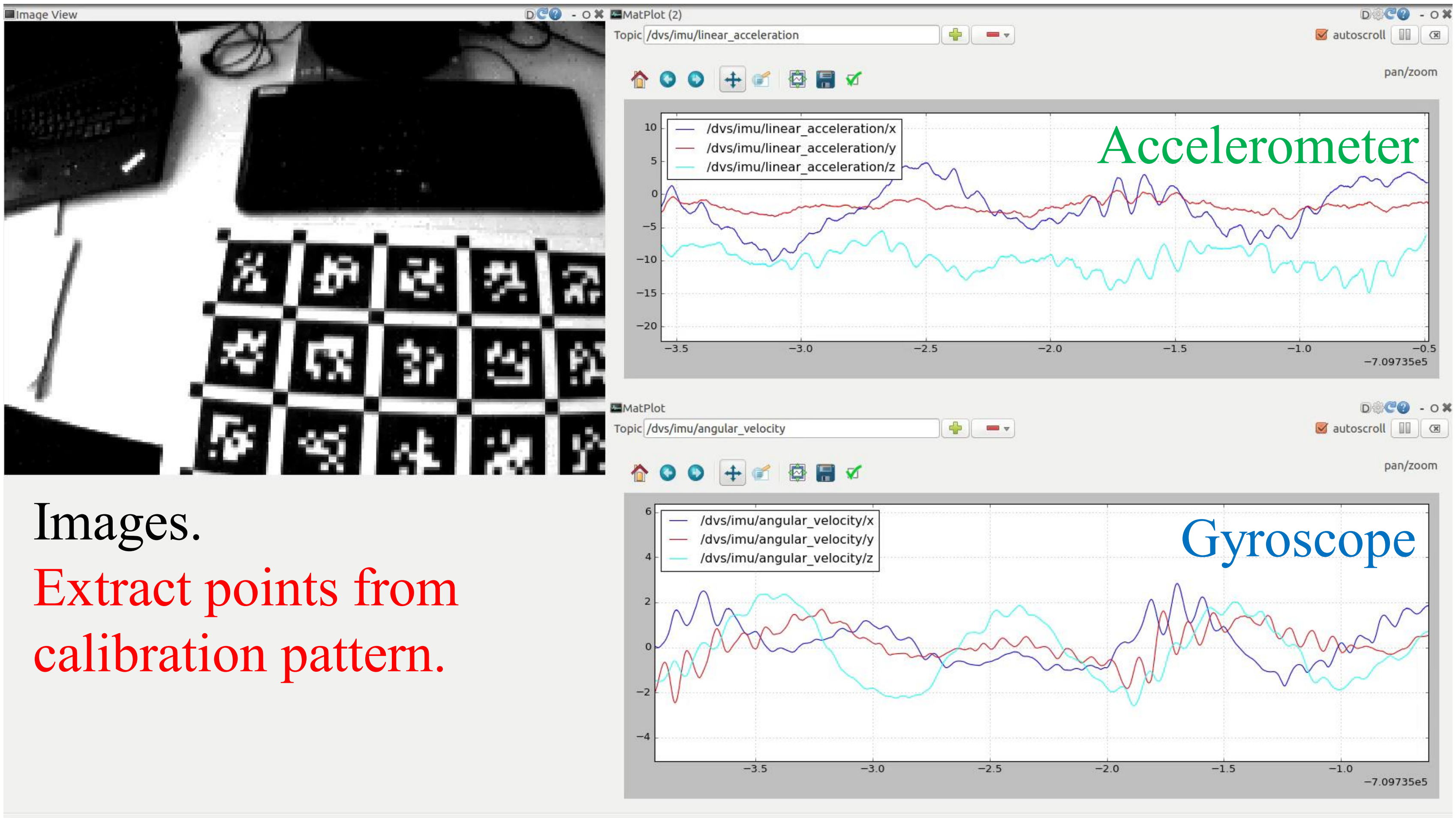




Furgale et al. "Unified Temporal and Spatial Calibration for Multi-Sensor Systems“, IROS’13.

Camera-IMU calibration - Example

- <u>Data acquisition</u>: Move the sensor in front of a static calibration pattern, exciting all degrees of freedom, and trying to make smooth motions.



Camera-IMU calibration

➢ <u>Approach</u>: Minimize a cost function (Furgale'13):

- $J(\theta) \coloneqq J_{feat} + J_{acc} + J_{gyro} + J_{bias_{acc}} + J_{bias_{gyro}}$

  (Feature reprojection Error) $\sum_k \left(a_{IMU}(t_k - t_d) - a_{Cam}(t_k)\right)^2$ $\sum_k \left(\omega_{IMU}(t_k - t_d) - \omega_{Cam}(t_k)\right)^2$ $\int \left\| \frac{db_{acc}}{dt}(u) \right\|^2 du$ $\int \left\| \frac{db_{gyro}}{dt}(u) \right\|^2 du$

- Unknowns: $T_{BC}, t_d, g_w, T_{WB}(t), b_{acc}(t), b_{gyro}(t)$

  - Gravity g_w, 6-DOF trajectory of the IMU $T_{WB}(t)$, 3-DOF biases of the IMU $b_{acc}(t), b_{gyro}(t)$

- Continuous-time modelling using splines for $T_{WB}(t)$, $b_{acc}(t)$, …

- Numerical solver: Levenberg-Marquardt (i.e., Gauss-Newton).

Furgale et al. "Unified Temporal and Spatial Calibration for Multi-Sensor Systems", IROS'13.

Camera-IMU calibration - Example

- Software solution: Kalibr (Furgale'13).
  - Generates a report after optimizing the cost function.

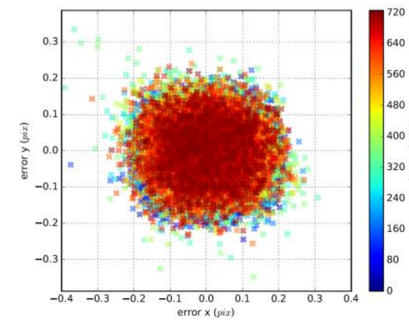
Residuals:
Reprojection error [px]: 0.0976 ± 0.051
Gyroscope error [rad/s]: 0.0167 ± 0.009
Accelerometer error [m/s^2]: 0.0595 ± 0.031

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.]]

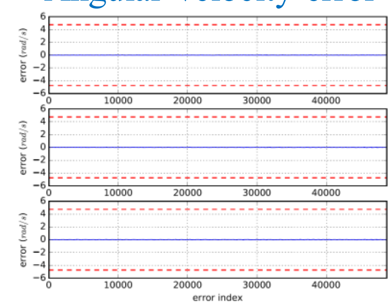
Time shift (delay *d*)
cam0 to imu0: [s] (t_imu = t_cam + shift)
0.00270636270255

Gravity vector in target coords: [m/s^2]
[0.04170719 -0.01000423 -9.80645621]

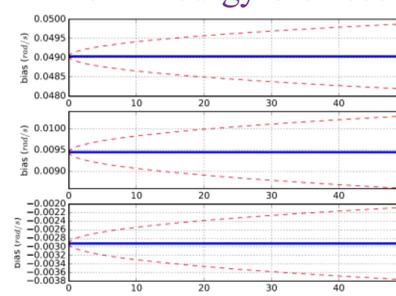
Reprojection error



Angular velocity error



Estimated gyro biases



Furgale et al. "Unified Temporal and Spatial Calibration for Multi-Sensor Systems", IROS'13.
https://github.com/ethz-asl/kalibr/wiki/camera-imu-calibration