Modern Robot State Estimation

10회차 통합 강의자료 | ESKF · Preintegration · MAP · MSCKF/VIO · LiDAR

APRL 강의자료

Modern Edition

DGIST 로봇및기계전자공학과

전체 목차 (1/3) — Part 1: 이론의 토대

Bayes 정리에서 출발해 SO(3)와 선형 KF로. 4회차 ESKF를 이해하기 위한 수학적 토대.

0회차 — 2025 필드 지형도
Filter vs Smoother 생태계 · ESKF가 사실상 표준이 된 이유 · 코스 로드맵

1회차 — Bayes filter, Gaussian
조건부 확률 / Bayes 정리 두 줄 유도 · prior × likelihood = posterior · posterior → 다음 턴의 prior (filter의 본질) · 가우시안 곱

2회차 — SO(3) 입문 + Jacobians
매니폴드 / tangent space · Exp / Log map · Right perturbation · Right Jacobian J_r, J_l · BCH 공식 · Adjoint

3회차 — 선형 Kalman Filter
KF 4가정 · 5줄 유도 (직관 + 행렬 미분) · Joseph form 수치 안정성 · KF 최적성 정리 · 톱니파(sawtooth) 분산 패턴

전체 목차 (2/3) — Part 2: ESKF 핵심

코스의 **심장**. 4회차 ESKF가 모든 후속 회차의 토대.

4회차 — ESKF 핵심 **(이 코스의 중심)**
EKF의 한계 · **Nominal / Error 상태 두 층 분리** · 15D 오차 벡터 $[\delta\boldsymbol{\phi}, \delta\boldsymbol{p}, \delta\boldsymbol{v}, \delta\boldsymbol{b}_g, \delta\boldsymbol{b}_a]$ · F matrix 유도 · Inject & Reset · 한 줄 요약

5회차 — IMU 물리 + ESKF Predict + Preintegration
IMU 측정 모델 (specific force) · 바이어스 임의 보행 · F/G 행렬 처음부터 유도 · OpenVINS 코드 매핑 · Forster 2017 preintegration · 6개 bias Jacobian

6회차 — 멀티센서 Update
측정 모델 일반론 · GPS H 행렬 · 휠 인코더 H (Adjoint 항등식) · 카메라 reprojection · Mahalanobis 검정 · 좌표계 변환

전체 목차 (3/3) — Part 3: ESKF 너머

2025년 현장 시스템(OpenVINS, FAST-LIO/LIVO)을 코드 단위로 읽어내는 능력.

7회차 — 관측가능성 + Factor Graph + MAP 최적화
Observability 정의 · NIS 튜닝 · MAP ≡ 비선형 최소제곱 · Gauss-Newton / LM · Schur complement · Marginalization · iSAM2

8회차 — MSCKF + Visual-Inertial Odometry
MSCKF 동기 · Stochastic cloning · Null-space projection (QR) · Feature triangulation · FEJ · OpenVINS 파이프라인 전체

9회차 — LiDAR 상태 추정 (FAST-LIO / FAST-LIVO)
ICP / LOAM / LIO-SAM · FAST-LIO IEKF on SO(3) · ikd-tree 동적 맵 · Point-to-plane Jacobian · FAST-LIVO/2

이 코스가 도달하는 한 줄

Bell & Cathey, IEEE TAC 1993

$$\text{ESKF} \subset \text{IEKF} \equiv \text{Gauss-Newton on MAP} \subset \text{Factor Graph}$$

같은 cost를 푸는 서로 다른 스케줄 — 필터와 스무더는 다른 알고리즘이 아니다.

0회차: 2025 필드 지형도

이 미니코스가 다루는 한 가지 질문

“내 로봇은 **지금** 어디에 있는가?”

이 미니코스가 다루는 한 가지 질문

"내 로봇은 **지금** 어디에 있는가?"

- GPS만? → 실내·터널·도심 협곡에서 동작 안 함
- 휠 인코더만? → 슬립 발생 순간 오차 누적
- IMU만? → 적분하면 **1분에 수백 미터** 발산

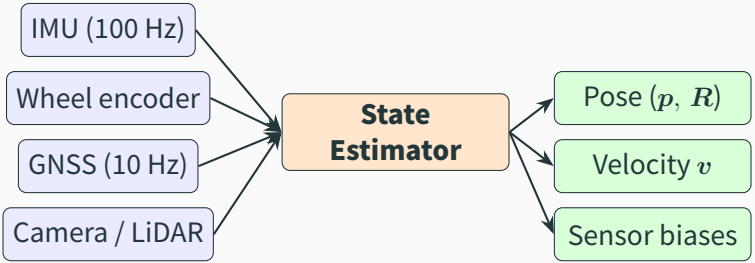
이 미니코스가 다루는 한 가지 질문

"내 로봇은 **지금** 어디에 있는가?"

- GPS만? → 실내·터널·도심 협곡에서 동작 안 함
- 휠 인코더만? → 슬립 발생 순간 오차 누적
- IMU만? → 적분하면 **1분에 수백 미터** 발산

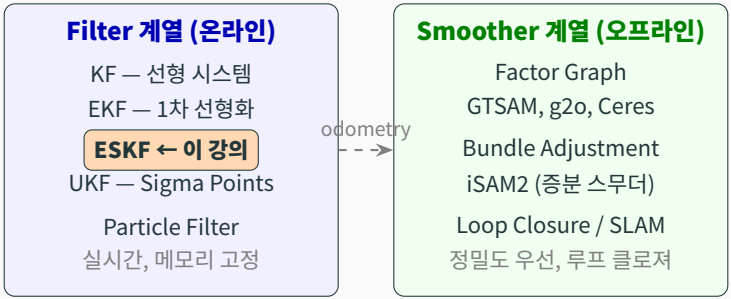
해답
여러 센서를 **확률적으로 결합**하는 알고리즘 — State Estimation

State estimation이 해결하는 것



각 센서의 장점은 취하고 단점은 다른 센서로 보완 — 이것이 fusion의 본질

2025년 기준 State Estimation 생태계 지형도



Filter vs. Smoother — 언제 뭘 쓰나

Filter (ESKF 등)

- 현재 시점까지의 측정만 사용
- 고정된 메모리, 고정된 연산량
- 실시간 제어 출력에 적합
- **실무 예**: 드론 자세 제어, 자율주행 odometry

Smoother (Factor Graph 등)

- 과거 측정을 재사용해 전체 최적화
- 메모리·연산이 시간에 비례 증가
- 루프 클로져, 정밀 지도 생성에 적합
- **실무 예**: SLAM, SfM, 오프라인 재처리

실제 시스템의 구조

Filter (ESKF) $\xrightarrow{\text{odometry}}$ **Smoother (Factor Graph)**

빠른 odometry가 SLAM의 초기값을 공급하는 2계층 구조가 표준

ESKF — 왜 지금 이것인가

Error-State Kalman Filter (ESKF)는 실시간 IMU 기반 fusion의 **사실상 표준**이다.

오픈소스 레퍼런스 구현 (ESKF/iEKF)

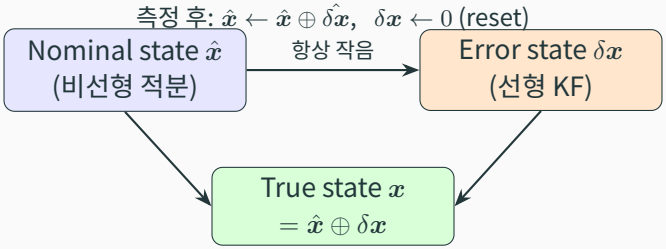
- **OpenVINS** (U. Delaware, 2020)
  Visual-Inertial Odometry, MSCKF + ESKF error-state
- **ROVIO** (ETH, 2017)
  VIO, IEKF on manifold
- **FAST-LIO2** (HKU MARS, 2022)
  LiDAR-IMU odometry, IEKF on SO(3)
- **FAST-LIVO2** (HKU MARS, 2024)
  LiDAR + Visual + Inertial, IEKF

인접 영역 (factor graph + preintegration): VINS-Mono, ORB-SLAM3 VI, LIO-SAM, Kalibr (batch).

UKF 대비 ESKF의 장점

- Sigma point 불필요
  연산량 동일 또는 적음
- 쿼터니언 부호 문제 **구조적 소멸**
- IMU 적분에 최적화된 설계
- 수많은 레퍼런스 코드

ESKF의 핵심 아이디어 — 미리보기



Error state $\delta \boldsymbol{x}$가 항상 작다는 것이 보장되므로 선형 KF를 안전하게 적용 가능

전체 로드맵 — Modern Robot State Estimation

강의	주제	핵심 내용
0	오리엔테이션	생태계 지형도, ESKF 위치 설정
1	Bayes 필터	확률, 가우시안, predict/update, posterior $\rightarrow$ next prior
2	SO(3) 입문 + Jacobians	매니폴드, Exp/Log, BCH, J_r / J_l, Adjoint
3	선형 KF	5줄 유도, Joseph form, 수치 안정성
4	**ESKF 핵심**	**nominal/error 분리, F matrix, inject & reset**
5	IMU + Predict + Preintegration	specific force, F/G 유도, Forster 2017 bias Jac.
6	멀티센서 update	GPS/wheel/camera H matrix 유도
7	관측가능성 + Factor Graph + MAP	NIS, GN/LM, Schur, Marginalization, iSAM2
8	MSCKF + VIO	Stochastic cloning, null-space proj., FEJ, OpenVINS
9	LiDAR 상태 추정	FAST-LIO/2, ikd-tree, FAST-LIVO/2

4강이 전체의 핵심. 1–3강은 수학적 토대, 7–9강은 ESKF를 넘어선 현대 robotics state estimation.

필요한 수학 배경

이미 알면 좋은 것

- 선형대수 (행렬 곱, 역행렬, 고유값)
- 확률론 기초 (가우시안, 조건부 확률)
- 미적분 (편미분, 연쇄 법칙)
- 쿼터니언 (개념 수준이면 충분)

이 강의에서 가르치는 것

- **Bayes 필터** (1강)
- **SO(3) 매니폴드** (2강)
- **Kalman Filter 유도** (3강)
- **ESKF 전체 구조** (4강)

이 강의의 약속
이 코스가 끝나면 OpenVINS의 `Propagator.cpp`/`UpdaterHelper.cpp`, 그리고 FAST-LIO2의 IEKF 구현을 한 줄씩 읽고 왜 이렇게 짰는지 설명할 수 있다.

참고 자료

- **Solà et al., “A micro Lie theory” (2018)** — SO(3), ESKF 바이블
  `arXiv:1812.01537`
- **Barfoot, “State Estimation for Robotics” (2017)** — 교과서 표준
- **Geneva et al., “OpenVINS” (ICRA 2020)** — MSCKF + ESKF error-state VIO
  `github.com/rpng/open_vins`
- **Qin et al., “VINS-Mono” (T-RO 2018)** — 드론 VIO (sliding-window factor graph + IMU preintegration)
- **Xu & Zhang, “FAST-LIO2” (T-RO 2022)** — iEKF on manifold, LiDAR-IMU 표준
- **Forster et al., “IMU Preintegration” (RSS 2015)** — Factor graph 브리지
- **Bell & Cathey (IEEE TAC 1993)** — IEKF ≡ Gauss-Newton 등치성 증명

다음 회차 예고: 1회차 — Bayes 정리에서 KF가 흘러나오는 모습

한 줄 질문

여러 센서를 “확률적으로 결합”한다는 것은 정확히 무슨 뜻인가?

다음 회차 예고: 1회차 — Bayes 정리에서 KF가 흘러나오는 모습

한 줄 질문

여러 센서를 “확률적으로 결합”한다는 것은 정확히 무슨 뜻인가?

1회차에서 답할 것

- Bayes 정리 두 줄 유도 — $p(\boldsymbol{x} \mid \boldsymbol{z}) \propto p(\boldsymbol{z} \mid \boldsymbol{x})\, p(\boldsymbol{x})$
- 가우시안 곱 = 평균/분산의 **가중합** → 이게 KF의 update 그 자체
- Posterior가 다음 턴의 prior가 되는 재귀 — “filter”라는 개념의 본질
- 톱니파(sawtooth) 분산 패턴 — predict가 늘리고 update가 줄임

다음 회차 예고: 1회차 — Bayes 정리에서 KF가 흘러나오는 모습

한 줄 질문

여러 센서를 “확률적으로 결합”한다는 것은 정확히 무슨 뜻인가?

1회차에서 답할 것

- Bayes 정리 두 줄 유도 — $p(\boldsymbol{x} \mid \boldsymbol{z}) \propto p(\boldsymbol{z} \mid \boldsymbol{x})\, p(\boldsymbol{x})$
- 가우시안 곱 = 평균/분산의 **가중합** → 이게 KF의 update 그 자체
- Posterior가 다음 턴의 prior가 되는 재귀 — “filter”라는 개념의 본질
- 톱니파(sawtooth) 분산 패턴 — predict가 늘리고 update가 줄임

1회차가 끝나면 KF의 “5줄”이 어디서 나오는지 머릿속에 그림이 그려진다.
그게 3회차 선형 KF, 4회차 ESKF의 기반이 된다.

1회차: Bayes filter, Gaussian

지난 시간 한 줄

어떤 단일 센서도 충분치 않다.
확률적으로 결합하는 알고리즘이 필요하다.

오늘의 목표:

- 확률, 가우시안, Bayes 정리 — 손에 익히기
- Bayes 정리에서 필연적으로 **predict / update** 두 단계가 흘러나오는 모습
- 1D 가우시안 곱 — 손으로 한 번, Python으로 한 번

용어 정리: 우리가 다루는 것은 모두 불확실하다

- state x: 알고 싶은 양 (위치, 속도, bias 등)
- measurement z: 센서가 알려주는 양
- 둘 다 확정값이 아니라 **확률분포**로 다룬다

용어 정리: 우리가 다루는 것은 모두 불확실하다

- state $\boldsymbol{x}$: 알고 싶은 양 (위치, 속도, bias 등)
- measurement $\boldsymbol{z}$: 센서가 알려주는 양
- 둘 다 확정값이 아니라 **확률분포**로 다룬다

이 코스의 표기
"$\boldsymbol{x}$의 belief (믿음)" = $p(\boldsymbol{x})$
"$\boldsymbol{z}$를 관측한 뒤 $\boldsymbol{x}$의 belief" = $p(\boldsymbol{x} \mid \boldsymbol{z})$

확률의 두 가지 해석

- **Frequentist:** “동전을 무한히 던지면 앞면 비율이 0.5”
- **Bayesian:** “이 동전이 앞면일 거라는 내 믿음이 0.5”

확률의 두 가지 해석

- **Frequentist:** “동전을 무한히 던지면 앞면 비율이 0.5”
- **Bayesian:** “이 동전이 앞면일 거라는 내 믿음이 0.5”

로봇 state estimation에서는?
당연히 **Bayesian**.
“지금 내 로봇 위치”를 무한히 반복할 수 없다.
우리는 단 한 번의 실현(이 로봇, 이 시점)에 대한 믿음을 다룬다.

조건부 확률 — 한 번에 정리

$$p(A \mid B) \;=\; \frac{p(A, B)}{p(B)}$$

- "B가 일어났다는 사실을 알 때, A의 확률"
- joint $p(A, B)$를 B의 단면으로 정규화한 것
- 이 정의 한 줄로부터 **Bayes 정리**와 **전확률 공식**이 모두 따라옴

Bayes 정리 — 단 두 줄로 유도

$p(A, B) = p(A \mid B)\, p(B) = p(B \mid A)\, p(A)$ 이므로

$$\boxed{p(A \mid B) \;=\; \frac{p(B \mid A)\, p(A)}{p(B)}}$$

Bayes 정리 — 단 두 줄로 유도

$p(A, B) = p(A \mid B)\, p(B) = p(B \mid A)\, p(A)$ 이므로

$$\boxed{p(A \mid B) \;=\; \frac{p(B \mid A)\, p(A)}{p(B)}}$$

state estimation 표기로 바꾸면

$$p(\boldsymbol{x} \mid \boldsymbol{z}) \;=\; \frac{p(\boldsymbol{z} \mid \boldsymbol{x})\, p(\boldsymbol{x})}{p(\boldsymbol{z})}$$

- $p(\boldsymbol{x})$: **prior** — 측정 전 믿음
- $p(\boldsymbol{z} \mid \boldsymbol{x})$: **likelihood** — 센서 모델
- $p(\boldsymbol{x} \mid \boldsymbol{z})$: **posterior** — 측정 후 믿음
- $p(\boldsymbol{z})$: 정규화 상수, 사실상 곱셈으로 취급

왜 비례식으로 충분한가

$$p(\boldsymbol{x} \mid \boldsymbol{z}) \;\propto\; p(\boldsymbol{z} \mid \boldsymbol{x})\, p(\boldsymbol{x})$$

- $p(\boldsymbol{z})$는 $\boldsymbol{x}$에 의존하지 않는다 — $\boldsymbol{x}$ 변화에 따라 모양이 바뀌지 않음
- 마지막에 분포 전체를 적분=1로 맞추면 자동 회복
- 그래서 모양만 보면 **prior** $\times$ **likelihood**가 곧 posterior

왜 비례식으로 충분한가

$$p(\boldsymbol{x} \mid \boldsymbol{z}) \;\propto\; p(\boldsymbol{z} \mid \boldsymbol{x})\, p(\boldsymbol{x})$$

- $p(\boldsymbol{z})$는 $\boldsymbol{x}$에 의존하지 않는다 — $\boldsymbol{x}$ 변화에 따라 모양이 바뀌지 않음
- 마지막에 분포 전체를 적분=1로 맞추면 자동 회복
- 그래서 모양만 보면 **prior** $\times$ **likelihood**가 곧 posterior

직관 한 줄
“사전 믿음을 측정값이 얼마나 그럴듯하게 만드는지로 가중한다.”

Bayes 정리 — 엔지니어링 관점의 두 가지 핵심

로봇 state estimation을 하는 사람에게 Bayes 정리는 두 줄로 요약된다.

#1 — Posterior는 prior와 observation의 가중 결합

$$\underbrace{p(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{posterior}} \propto \underbrace{p(\boldsymbol{z} \mid \boldsymbol{x})}_{\text{observation}} \cdot \underbrace{p(\boldsymbol{x})}_{\text{prior}}$$

가우시안이 되면 이것은 곧 **평균의 가중합**, 가중치는 정밀도(분산의 역수)이다.
KF의 갱신식 $\hat{\boldsymbol{x}}^{+} = \hat{\boldsymbol{x}}^{-} + \boldsymbol{K}(\boldsymbol{z} - \boldsymbol{H}\hat{\boldsymbol{x}}^{-})$가 정확히 이 형태다.

Bayes 정리 — 엔지니어링 관점의 두 가지 핵심

로봇 state estimation을 하는 사람에게 Bayes 정리는 두 줄로 요약된다.

#1 — Posterior는 prior와 observation의 가중 결합

$$\underbrace{p(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{posterior}} \propto \underbrace{p(\boldsymbol{z} \mid \boldsymbol{x})}_{\text{observation}} \cdot \underbrace{p(\boldsymbol{x})}_{\text{prior}}$$

가우시안이 되면 이것은 곧 **평균의 가중합**, 가중치는 정밀도(분산의 역수)이다.
KF의 갱신식 $\hat{\boldsymbol{x}}^{+} = \hat{\boldsymbol{x}}^{-} + \boldsymbol{K}(\boldsymbol{z} - \boldsymbol{H}\hat{\boldsymbol{x}}^{-})$가 정확히 이 형태다.

#2 — Posterior는 다음 턴의 prior가 된다 (이게 더 중요)

$$p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t}) \xrightarrow{\text{predict}} p(\boldsymbol{x}_{t+1} \mid \boldsymbol{z}_{1:t}) \xrightarrow{\text{update}} p(\boldsymbol{x}_{t+1} \mid \boldsymbol{z}_{1:t+1})$$

한 번만 결합하는 것이 아니라 **무한 재귀**로 결합한다.
이 재귀가 "filter"라는 개념의 본질 — 시간을 따라 belief가 흐른다.

왜 하필 가우시안인가 — 닫힌 형식의 선택

- **중심극한정리**: 독립 잡음의 합 → 가우시안 (실제 센서 잡음의 좋은 근사)
- **닫힌 형식**: 가우시안끼리 곱해도 또 가우시안
- **선형 변환**에 닫혀있음: $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu},\, \boldsymbol{\Sigma})$이면 $\boldsymbol{A}\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{A}\boldsymbol{\mu},\, \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^{\top}\right)$
- 평균 + 공분산 두 모멘트만으로 완전히 기술 — **메모리 효율**

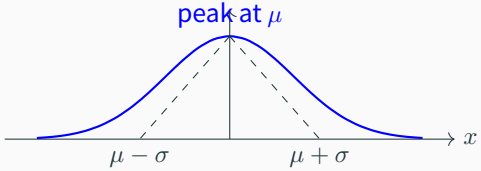
왜 하필 가우시안인가 — 닫힌 형식의 선택

- **중심극한정리**: 독립 잡음의 합 → 가우시안 (실제 센서 잡음의 좋은 근사)
- **닫힌 형식**: 가우시안끼리 곱해도 또 가우시안
- **선형 변환**에 닫혀있음: $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu},\, \boldsymbol{\Sigma})$이면 $\boldsymbol{A}\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{A}\boldsymbol{\mu},\, \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^{\top}\right)$
- 평균 + 공분산 두 모멘트만으로 완전히 기술 — **메모리 효율**

주의
가우시안은 닫힌 형식이 가능한 선택이지 유일한 선택이 아니다.
비가우시안 문제(다봉 분포, 심한 비선형) → **Particle filter**가 일반 해.
이 코스는 가우시안으로 제한하되, 그 한계를 명확히 인식하고 넘어간다.

1D 가우시안 — 식과 그림

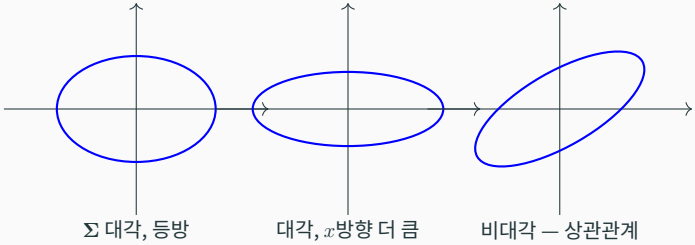
$$\mathcal{N}(x;\mu,\sigma^2) \;=\; \frac{1}{\sqrt{2\pi}\,\sigma}\,\exp\!\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



σ는 불확실성 폭이고, 작아질수록 분포가 뾰족해진다.

다변수 가우시안 — 공분산이 모양을 정한다

$$\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) \;=\; \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}}\exp\!\Big(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big)$$



비대각 항(off-diagonal)이 KF의 진짜 힘이 어디서 오는지를 결정한다.

핵심 보조 정리: 가우시안 $\times$ 가우시안

두 1D 가우시안 $\mathcal{N}(x;\mu_1,\sigma_1^2), \mathcal{N}(x;\mu_2,\sigma_2^2)$의 곱은 다시 가우시안:

$$\mu = \frac{\mu_1/\sigma_1^2 + \mu_2/\sigma_2^2}{1/\sigma_1^2 + 1/\sigma_2^2}, \qquad \frac{1}{\sigma^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}$$

핵심 보조 정리: 가우시안 $\times$ 가우시안

두 1D 가우시안 $\mathcal{N}(x;\mu_1,\sigma_1^2), \mathcal{N}(x;\mu_2,\sigma_2^2)$의 곱은 다시 가우시안:

$$\mu = \frac{\mu_1/\sigma_1^2 + \mu_2/\sigma_2^2}{1/\sigma_1^2 + 1/\sigma_2^2}, \qquad \frac{1}{\sigma^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}$$

- 평균은 **정밀도(=$1/\sigma^2$) 가중 평균**
- 분산은 **역수의 합**으로 결합 → 항상 둘 중 작은 값보다 더 작다
- 즉 두 정보를 합치면 불확실성이 줄어든다

가우시안 곱 — 손으로 유도 (1)

$$\mathcal{N}(\mu_1, \sigma_1^2)\,\mathcal{N}(\mu_2, \sigma_2^2) \propto \exp\!\left(-\tfrac{(x-\mu_1)^2}{2\sigma_1^2} - \tfrac{(x-\mu_2)^2}{2\sigma_2^2}\right)$$

가우시안 곱 — 손으로 유도 (1)

$$\mathcal{N}(\mu_1,\sigma_1^2)\,\mathcal{N}(\mu_2,\sigma_2^2) \propto \exp\!\left(-\tfrac{(x-\mu_1)^2}{2\sigma_1^2}-\tfrac{(x-\mu_2)^2}{2\sigma_2^2}\right)$$

지수 안을 x에 대해 정리:

$$-\tfrac{1}{2}\left[\left(\tfrac{1}{\sigma_1^2}+\tfrac{1}{\sigma_2^2}\right)x^2-2\left(\tfrac{\mu_1}{\sigma_1^2}+\tfrac{\mu_2}{\sigma_2^2}\right)x+\text{const}\right]$$

가우시안 곱 — 손으로 유도 (1)

$$\mathcal{N}(\mu_1, \sigma_1^2)\,\mathcal{N}(\mu_2, \sigma_2^2) \propto \exp\!\left(-\tfrac{(x-\mu_1)^2}{2\sigma_1^2} - \tfrac{(x-\mu_2)^2}{2\sigma_2^2}\right)$$

지수 안을 x에 대해 정리:

$$-\tfrac{1}{2}\left[\left(\tfrac{1}{\sigma_1^2} + \tfrac{1}{\sigma_2^2}\right)x^2 - 2\left(\tfrac{\mu_1}{\sigma_1^2} + \tfrac{\mu_2}{\sigma_2^2}\right)x + \text{const}\right]$$

새 분산: $1/\sigma^2 = 1/\sigma_1^2 + 1/\sigma_2^2$

새 평균: $\mu = \sigma^2 \cdot (\mu_1/\sigma_1^2 + \mu_2/\sigma_2^2)$

가우시안 곱 — 손으로 유도 (2)

대입하면 μ가 정밀도 가중평균 형태로 단순화:

$$\mu = \frac{\mu_1/\sigma_1^2 + \mu_2/\sigma_2^2}{1/\sigma_1^2 + 1/\sigma_2^2}$$

가우시안 곱 — 손으로 유도 (2)

대입하면 μ가 정밀도 가중평균 형태로 단순화:

$$\mu = \frac{\mu_1/\sigma_1^2 + \mu_2/\sigma_2^2}{1/\sigma_1^2 + 1/\sigma_2^2}$$

식의 의미를 다시 보면

- $\sigma_1 \to 0$: 첫 측정이 완벽 → $\mu \to \mu_1$ (그 값을 그대로 받음)
- $\sigma_1 \to \infty$: 첫 측정이 무의미 → $\mu \to \mu_2$ (둘째 값만 사용)
- 그 사이는 매끄럽게 보간

같은 식, KF 게인 형태로 다시 쓰기

$K = \dfrac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}$ 라 하면

$$\mu = \mu_1 + K(\mu_2 - \mu_1), \qquad \sigma^2 = (1 - K)\,\sigma_1^2$$

같은 식, KF 게인 형태로 다시 쓰기

$K = \dfrac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}$ 라 하면

$$\mu = \mu_1 + K(\mu_2 - \mu_1), \qquad \sigma^2 = (1 - K)\,\sigma_1^2$$

이게 바로 1D Kalman update

- μ_1, σ_1^2: prior
- μ_2, σ_2^2: 측정
- K: **Kalman gain** — "측정을 얼마나 신뢰할지"
- $(\mu_2 - \mu_1)$: **innovation** — 측정과 예측의 차이

3회차에서 이 형태가 다차원으로 그대로 일반화된다.

이제 Bayes 정리로 돌아가서 — 시간을 끌어들이자

로봇은 시계열이다. 시간 k의 state $\boldsymbol{x}_k$를 추정하고 싶다.

- 두 종류의 정보:
  1. **운동 모델** $p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})$ — "이전 상태에서 어떻게 움직였을까"
  2. **측정 모델** $p(\boldsymbol{z}_k \mid \boldsymbol{x}_k)$ — "이 상태일 때 센서가 뭘 보여줄까"

이제 Bayes 정리로 돌아가서 — 시간을 끌어들이자

로봇은 시계열이다. 시간 k의 state $\boldsymbol{x}_k$를 추정하고 싶다.

- 두 종류의 정보:
  1. **운동 모델** $p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})$ — "이전 상태에서 어떻게 움직였을까"
  2. **측정 모델** $p(\boldsymbol{z}_k \mid \boldsymbol{x}_k)$ — "이 상태일 때 센서가 뭘 보여줄까"

Bayes 정리를 시간 순으로 풀어 적용하면 두 단계가 자연스럽게 분리된다.

Bayes filter — predict 단계 유도

$\boldsymbol{x}_k$의 prior를 이전 posterior로부터 만든다 (전확률 공식):

$$p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k-1}) \;=\; \int p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})\, p(\boldsymbol{x}_{k-1} \mid \boldsymbol{z}_{1:k-1})\, d\boldsymbol{x}_{k-1}$$

- 운동 모델 $p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})$로 “이전 belief를 미래로 옮긴다”
- 측정값은 아직 안 들어왔다 — 그래서 **predict**
- 결과로 belief가 퍼진다(불확실성 증가) — 직관과 일치

Bayes filter — update 단계 유도

새 측정 $\boldsymbol{z}_k$가 들어오면 Bayes 정리로 갱신:

$$p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k}) \;\propto\; p(\boldsymbol{z}_k \mid \boldsymbol{x}_k)\, p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k-1})$$

- likelihood $\times$ prior $\rightarrow$ posterior
- 측정이 좁히는 효과 — 불확실성 감소

Bayes filter — update 단계 유도

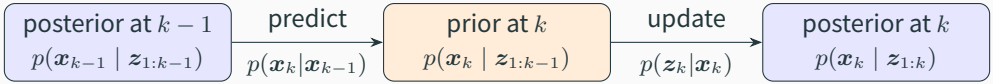
새 측정 z_k가 들어오면 Bayes 정리로 갱신:

$$p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k}) \;\propto\; p(\boldsymbol{z}_k \mid \boldsymbol{x}_k)\, p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k-1})$$

- likelihood $\times$ prior $\rightarrow$ posterior
- 측정이 좁히는 효과 — 불확실성 감소

핵심 메시지
predict는 운동 모델로 시간을 흘려보내고, **update**는 측정으로 분포를 좁힌다.
이 두 단계의 분리는 선택이 아니라 Bayes 정리의 필연이다.

Bayes filter — 한 슬라이드로



- 매 시간 스텝마다 두 박스를 번갈아 통과
- ESKF의 100 Hz predict + 비동기 update가 정확히 이 구조

Bayes filter는 완벽하지만 풀 수 없다

- predict의 적분 $\int \cdots d\boldsymbol{x}_{k-1}$ — 일반적으로 닫힌 해 없음
- update의 정규화 $p(\boldsymbol{z})$ — 일반적으로 닫힌 해 없음
- state가 연속, 분포 임의 모양이면 **이론상 가능**하지만 컴퓨터로 못 풂

Bayes filter는 완벽하지만 풀 수 없다

- predict의 적분 $\int \cdots d\boldsymbol{x}_{k-1}$ — 일반적으로 닫힌 해 없음
- update의 정규화 $p(\boldsymbol{z})$ — 일반적으로 닫힌 해 없음
- state가 연속, 분포 임의 모양이면 **이론상 가능**하지만 컴퓨터로 못 풂

돌파구
분포를 가우시안으로 가정하면 두 단계 모두 닫힌 해가 나온다.
이게 **Kalman Filter**.

KF는 Bayes filter + 가우시안 가정

- prior, posterior를 모두 가우시안으로 표현
- 운동 모델, 측정 모델도 선형 + 가우시안 잡음으로 가정
- 그러면 두 단계가 **평균과 공분산의 업데이트**로 환원됨
- 식 5줄로 끝 — 3회차에서 정식 유도

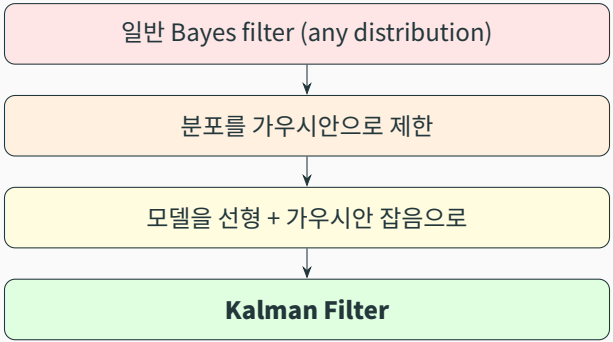
KF는 Bayes filter + 가우시안 가정

- prior, posterior를 모두 가우시안으로 표현
- 운동 모델, 측정 모델도 선형 + 가우시안 잡음으로 가정
- 그러면 두 단계가 **평균과 공분산의 업데이트**로 환원됨
- 식 5줄로 끝 — 3회차에서 정식 유도

비선형이면?
ESKF — 4회차

비-가우시안이면?
Particle filter (이 코스 범위 밖)

Bayes filter의 가정을 KF로 좁히는 사다리



- 우리는 사다리를 한 칸 한 칸 내려가며 왜 그런 가정이 필요한지 추적할 것
- 4회차에서 “비선형”으로 한 단 위로 올라간다 (ESKF)

사고실험 — 직선 복도 + 점자블럭

- state: x (직선 위 위치)
- 운동 모델: 한 발 옮기면 $x \leftarrow x + 0.7 + w, w \sim \mathcal{N}(0, 0.05^2)$
- 측정 모델: 점자가 있는 정확한 1m 지점에서 $z = x + v, v \sim \mathcal{N}(0, 0.02^2)$

사고실험 — 직선 복도 + 점자블럭

- state: x (직선 위 위치)
- 운동 모델: 한 발 옮기면 $x \leftarrow x + 0.7 + w, w \sim \mathcal{N}(0, 0.05^2)$
- 측정 모델: 점자가 있는 정확한 1m 지점에서 $z = x + v, v \sim \mathcal{N}(0, 0.02^2)$

- predict: 매 발걸음마다 σ가 커진다
- update: 점자 측정이 들어오면 가우시안 곱 → σ 다시 작아진다
- 두 효과의 균형이 지속 가능한 추정을 만든다

update — 가우시안 두 개의 곱으로 쓰기

prior (predict 결과): $\mathcal{N}(x;\ 70,\ 0.25)$ at 100보
measurement (점자 100번째): $\mathcal{N}(x;\ 100,\ 0.0004)$

update — 가우시안 두 개의 곱으로 쓰기

prior (predict 결과): $\mathcal{N}(x;\ 70,\ 0.25)$ at 100보
measurement (점자 100번째): $\mathcal{N}(x;\ 100,\ 0.0004)$ 곱하면:

$$\frac{1}{\sigma^2} = \frac{1}{0.25} + \frac{1}{0.0004} \approx 2504 \quad \Rightarrow \quad \sigma \approx 0.02$$

$$\mu = \frac{70/0.25 + 100/0.0004}{2504} \approx 99.99$$

- 결과: 분산은 측정의 분산과 거의 같음
- 직관: 정확한 측정 한 번이 prior를 거의 덮어쓴다

왜 우리는 공분산이라는 행렬을 굳이 쓰는가

- state가 단일 변수라면 분산만 있으면 됨
- 그러나 state가 여러 차원이고 **서로 상관**되어 있으면?

왜 우리는 공분산이라는 행렬을 굳이 쓰는가

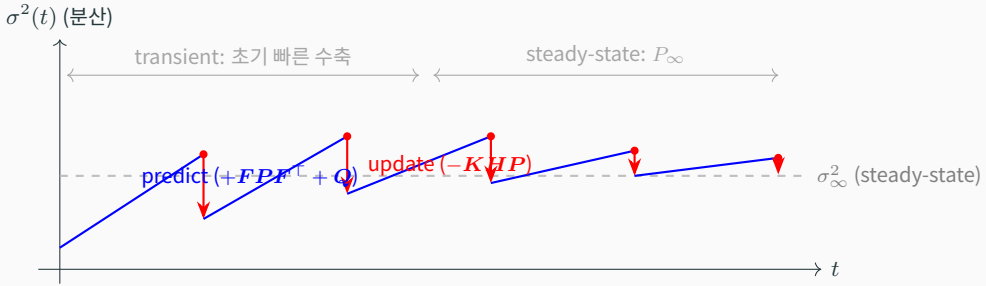
- state가 단일 변수라면 분산만 있으면 됨
- 그러나 state가 여러 차원이고 **서로 상관**되어 있으면?

예: 위치 x와 속도 v

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \rho\,\sigma_x\sigma_v \\ \rho\,\sigma_x\sigma_v & \sigma_v^2 \end{pmatrix}$$

- $\rho > 0$: “위치를 더 안다”는 정보가 들어오면 속도 추정도 좁아진다
- 이 상호작용이 KF의 진짜 힘 — **한 변수의 측정이 다른 변수까지 좁힌다**
- IMU bias 추정이 자동으로 되는 이유도 똑같음 (5회차에서 적용)

predict / update 반복 — 분산이 어떻게 움직이나 (sawtooth)



- **파란선**: predict가 분산을 늘림 ($P \mapsto FPF^\top + Q$)
- **빨간 화살**: update가 분산을 줄임 ($P \mapsto (I - KH)P$)
- **톱니파(sawtooth)**: 매 사이클 predict↑ → update↓ 반복

- **Steady-state**: 관측가능 시스템에서 $P \to P_\infty$ 수렴 (DARE 해)
- **ESKF 실제 비율**: predict 100–400 Hz, GPS 1–10 Hz, camera 10–30 Hz
- update 빈도가 낮을수록 톱니의 진폭이 커짐

잠깐 — likelihood vs measurement noise

- likelihood $p(\boldsymbol{z} \mid \boldsymbol{x})$ — 어떤 state일 때 어떤 측정이 나올 확률
- measurement noise $\boldsymbol{v} \sim \mathcal{N}(0,\ \boldsymbol{R})$ — 그 확률을 만드는 잡음
- 가우시안 가정에서 둘은 사실상 같은 객체의 두 표현

잠깐 — likelihood vs measurement noise

- likelihood $p(\boldsymbol{z} \mid \boldsymbol{x})$ — 어떤 state일 때 어떤 측정이 나올 확률
- measurement noise $\boldsymbol{v} \sim \mathcal{N}(0,\ \boldsymbol{R})$ — 그 확률을 만드는 잡음
- 가우시안 가정에서 둘은 사실상 같은 객체의 두 표현

practical 메모
$\boldsymbol{R}$를 너무 작게 잡으면 (“내 센서는 매우 정확해!”) → 노이즈를 진실로 받아들여 추정이 진동
$\boldsymbol{R}$를 너무 크게 잡으면 → 측정을 거의 무시 → 사실상 dead reckoning

1D Bayes filter — Python 구현


```python
import numpy as np

def predict(mu, var, u, q):
    # x_k = x_{k-1} + u + w,  w ~ N(0, q)
    return mu + u, var + q

def update(mu, var, z, r):
    # z = x + v,  v ~ N(0, r)
    K = var / (var + r)        # Kalman gain (1D)
    return mu + K*(z - mu), (1 - K)*var

mu, var = 0.0, 1.0
u, q, r = 0.7, 0.05**2, 0.02**2
for k in range(1, 101):
    mu, var = predict(mu, var, u, q)
    if k % 10 == 0:            # measurement every 10 steps
        mu, var = update(mu, var, k * 1.0, r)
```

- predict: 분산 커짐, update: 분산 줄어듦
- 측정 주기를 바꾸면($k\%20, k\%50$) 분산이 어떻게 달라지는가?

이번 회차 핵심 다섯 개

1. Bayes 정리 한 줄에서 predict/update가 필연으로 따라온다
2. 가우시안 곱 = 정밀도 가중평균 + 정밀도 합
3. 같은 식을 다시 쓰면 1D Kalman update의 표준 형태
4. 가우시안은 닫힌 형식이 가능한 선택 — 비가우시안은 Particle filter
5. 공분산의 비대각이 한 변수의 측정으로 다른 변수까지 좁히게 만든다

다음 시간 예고 — 2회차: SO(3)와 회전

회전이라는 매니폴드

- Euler angle의 특이점(Gimbal lock) — 왜 위험한가
- 쿼터니언: 더 낫지만 제약($|\mathbf{q}| = 1$)이 있다
- **SO(3)**: 회전 전체가 이루는 공간, 그 tangent space = $\mathfrak{so}(3) \cong \mathbb{R}^3$
- Exp / Log map: 회전 벡터와 회전 행렬을 오가는 다리
- 왜 이게 ESKF의 핵심인가 — error state가 $\mathfrak{so}(3)$에 산다

미니 예고

노트북을 x축 $90°$ 후 z축 $90°$ = z축 $90°$ 후 x축 $90°$?
순서를 바꾸면 결과가 다르다 — 이게 SO(3)가 벡터 공간이 아닌 이유.

2회차: SO(3) 입문

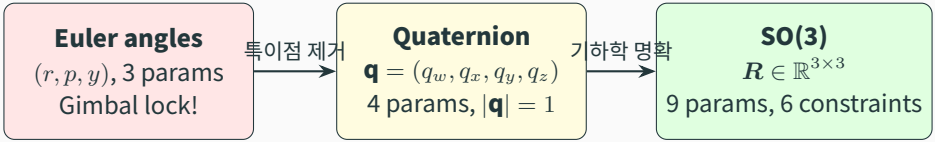
지난 시간 한 줄

Bayes filter는 Predict + Update.
가우시안이면 닫힌 형식으로 풀린다 $\rightarrow$ **KF**.

오늘의 목표:

- 회전 표현의 생태계 — 왜 쿼터니언도 완벽하지 않은가
- **SO(3)**: 회전이 이루는 공간의 기하학적 구조
- **tangent space** $\cong \mathbb{R}^3$: 작은 회전은 벡터다
- Exp / Log map: 회전 벡터 $\leftrightarrow$ 회전 행렬
- 왜 이것이 **ESKF의 핵심 기반**인가

회전 표현의 세 가지 선택지



- 어떤 표현도 “공짜로” 완벽하지 않음 — 표현마다 장단점이 있다
- ESKF는 **쿼터니언(구현)** + **SO(3) 기하학(이론)**을 결합

Euler angle의 치명적 결함 — Gimbal lock

Roll-Pitch-Yaw 표현에서 pitch = 90°이면:

- x축 회전과 z축 회전이 같은 방향을 가리킴 $\rightarrow$ DOF 소실
- 미분을 구하면 행렬식 $= 0 \rightarrow$ **Jacobian 특이**
- KF의 F, H matrix가 수치적으로 폭발
- 실제 사례: 아폴로 11호 짐벌 잠김, 비행기 자세 추정 오류

Euler angle의 치명적 결함 — Gimbal lock

Roll-Pitch-Yaw 표현에서 pitch = 90°이면:

- x축 회전과 z축 회전이 같은 방향을 가리킴 $\rightarrow$ DOF 소실
- 미분을 구하면 행렬식 $= 0 \rightarrow$ **Jacobian 특이**
- KF의 F, H matrix가 수치적으로 폭발
- 실제 사례: 아폴로 11호 짐벌 잠김, 비행기 자세 추정 오류

결론
회전을 미분 가능한 형태로 다루는 estimation 문제에서
Euler angle은 원천적으로 사용할 수 없다.

쿼터니언 — 더 낫지만 제약이 있다

$$\mathbf{q} = (q_w,\ q_x,\ q_y,\ q_z), \quad |\mathbf{q}| = 1 \text{ (단위 쿼터니언)}$$

- **장점**: Gimbal lock 없음, 회전 합성은 단순 곱
- **회전 표현**: 축 $\hat{\mathbf{n}}$, 각 $\theta \Rightarrow \mathbf{q} = (\cos\frac{\theta}{2},\ \hat{\mathbf{n}}\sin\frac{\theta}{2})$
- **단점**: $\mathbf{q}$와 $-\mathbf{q}$가 같은 회전 표현 (double-cover)

표기 주의: 이 코스는 **Hamilton 컨벤션** (q_w, q_x, q_y, q_z) 사용 — Eigen, GTSAM, ROS `tf2`와 일치.
OpenVINS는 **JPL 컨벤션** (q_x, q_y, q_z, q_w) 사용 — 코드 인용 시 변환 필요 (5/8회차에서 명시).

쿼터니언 — 더 낫지만 제약이 있다

$$\mathbf{q} = (q_w,\, q_x,\, q_y,\, q_z), \quad |\mathbf{q}| = 1 \text{ (단위 쿼터니언)}$$

- **장점**: Gimbal lock 없음, 회전 합성은 단순 곱
- **회전 표현**: 축 $\hat{\mathbf{n}}$, 각 $\theta \Rightarrow \mathbf{q} = (\cos\frac{\theta}{2},\ \hat{\mathbf{n}}\sin\frac{\theta}{2})$
- **단점**: $\mathbf{q}$와 $-\mathbf{q}$가 같은 회전 표현 (double-cover)

표기 주의: 이 코스는 **Hamilton 컨벤션** (q_w, q_x, q_y, q_z) 사용 — Eigen, GTSAM, ROS tf2와 일치.
OpenVINS는 **JPL 컨벤션** (q_x, q_y, q_z, q_w) 사용 — 코드 인용 시 변환 필요 (5/8회차에서 명시).

KF에서 쿼터니언의 문제
KF는 상태를 벡터로 다룬다.
$\hat{\mathbf{q}} + K\nu$로 갱신 $\rightarrow |\mathbf{q}| \neq 1$이 될 수 있다.
단순 renormalize를 하면 공분산이 잘못된 방향을 가리킨다.

SO(3) — 3D 회전의 집합 그 자체

$$\mathrm{SO}(3) \;=\; \left\{ \boldsymbol{R} \in \mathbb{R}^{3\times 3} \;\middle|\; \boldsymbol{R}^{\top}\boldsymbol{R} = \boldsymbol{I},\ \det(\boldsymbol{R}) = +1 \right\}$$

- 6개의 제약으로 $\mathbb{R}^9$ 안에서 **3차원 면(manifold)**이 된다
- 이 면 위의 각 점 = 하나의 3D 회전
- 두 회전의 합성: 행렬 곱 $\boldsymbol{R}_1\boldsymbol{R}_2$ (닫혀있음)
- 항등원: $\boldsymbol{I}$, 역원: $\boldsymbol{R}^{\top}$
- **군(group)** + 미분 가능한 면 $\rightarrow$ **Lie group**

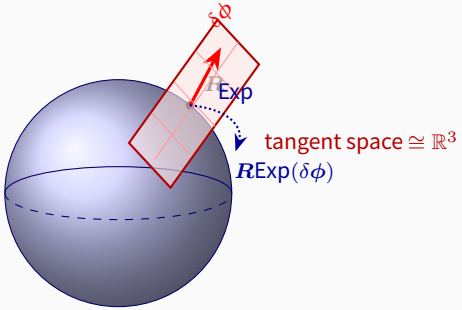
매니폴드 직관 — 구면으로 생각하기

지구 표면을 생각하자:

- 전체 공간: $\mathbb{R}^3$ (3D)
- 지구 표면: $\mathbb{R}^3$에 박힌 **2D 면** $= S^2$
- 한 점에서 아주 작은 이동은 **평면**처럼 느껴진다
- 그 "국소 평면" = **tangent space**

SO(3)도 마찬가지:

- $\mathbb{R}^9$ 안의 3D 면
- 한 회전 $\boldsymbol{R}$ 근방의 작은 변화
  $\rightarrow$ tangent space $\cong \mathbb{R}^3$



- 빨간 평면 = $\boldsymbol{R}$ 점에서의 tangent space
- 점선 곡선 = Exp map (평면 → 구면)

Lie algebra $\mathfrak{so}(3)$ — tangent space at identity

$\boldsymbol{R} = \boldsymbol{I}$ 근방에서 SO(3)의 tangent space를 **Lie algebra**라 부른다:

$$\mathfrak{so}(3) \;=\; \left\{ \boldsymbol{\Phi} \in \mathbb{R}^{3\times 3} \;\middle|\; \boldsymbol{\Phi} + \boldsymbol{\Phi}^{\top} = \boldsymbol{0} \right\} \quad \text{(반대칭 행렬)}$$

Lie algebra $\mathfrak{so}(3)$ — tangent space at identity

$\boldsymbol{R} = \boldsymbol{I}$ 근방에서 SO(3)의 tangent space를 **Lie algebra**라 부른다:

$$\mathfrak{so}(3) \;=\; \left\{ \boldsymbol{\Phi} \in \mathbb{R}^{3\times 3} \;\middle|\; \boldsymbol{\Phi} + \boldsymbol{\Phi}^{\top} = \boldsymbol{0} \right\} \quad \text{(반대칭 행렬)}$$

반대칭 행렬은 **3개의 자유 파라미터**만 가진다:

$$[\boldsymbol{\phi}]_{\times} \;=\; \begin{pmatrix} 0 & -\phi_z & \phi_y \\ \phi_z & 0 & -\phi_x \\ -\phi_y & \phi_x & 0 \end{pmatrix}, \quad \boldsymbol{\phi} = (\phi_x, \phi_y, \phi_z)^{\top} \in \mathbb{R}^3$$

Lie algebra $\mathfrak{so}(3)$ — tangent space at identity

$\boldsymbol{R} = \boldsymbol{I}$ 근방에서 SO(3)의 tangent space를 **Lie algebra**라 부른다:

$$\mathfrak{so}(3) \;=\; \left\{ \boldsymbol{\Phi} \in \mathbb{R}^{3\times 3} \;\middle|\; \boldsymbol{\Phi} + \boldsymbol{\Phi}^{\top} = \boldsymbol{0} \right\} \quad \text{(반대칭 행렬)}$$

반대칭 행렬은 **3개의 자유 파라미터**만 가진다:

$$[\boldsymbol{\phi}]_{\times} \;=\; \begin{pmatrix} 0 & -\phi_z & \phi_y \\ \phi_z & 0 & -\phi_x \\ -\phi_y & \phi_x & 0 \end{pmatrix}, \quad \boldsymbol{\phi} = (\phi_x, \phi_y, \phi_z)^{\top} \in \mathbb{R}^3$$

핵심
$\mathfrak{so}(3) \cong \mathbb{R}^3$. 작은 회전 $\delta\boldsymbol{R} \approx \boldsymbol{I}$는 **3D 벡터** $\boldsymbol{\phi}$로 기술된다.
ESKF에서 rotation error state $\delta\boldsymbol{\phi} \in \mathbb{R}^3$가 등장하는 이유가 바로 이것.

Exponential map — 벡터를 회전으로

$$\mathrm{Exp}(\boldsymbol{\phi}) \;:=\; e^{[\boldsymbol{\phi}]_\times} \;=\; \boldsymbol{I} + \frac{\sin\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|}\,[\boldsymbol{\phi}]_\times + \frac{1-\cos\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^2}\,[\boldsymbol{\phi}]_\times^2$$

(Rodrigues 공식)

- 입력: $\boldsymbol{\phi} \in \mathbb{R}^3$ (방향 = 회전축, 크기 = 각도)
- 출력: $\boldsymbol{R} \in \mathrm{SO}(3)$
- $\|\boldsymbol{\phi}\| \to 0$이면 $\mathrm{Exp}(\boldsymbol{\phi}) \approx \boldsymbol{I} + [\boldsymbol{\phi}]_\times$ (1차 근사)

Exponential map — 벡터를 회전으로

$$\mathrm{Exp}(\boldsymbol{\phi}) \;:=\; e^{[\boldsymbol{\phi}]_\times} \;=\; \boldsymbol{I} + \frac{\sin\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|}\,[\boldsymbol{\phi}]_\times + \frac{1-\cos\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^2}\,[\boldsymbol{\phi}]_\times^2$$

(Rodrigues 공식)

- 입력: $\boldsymbol{\phi} \in \mathbb{R}^3$ (방향 = 회전축, 크기 = 각도)
- 출력: $\boldsymbol{R} \in \mathrm{SO}(3)$
- $\|\boldsymbol{\phi}\| \to 0$이면 $\mathrm{Exp}(\boldsymbol{\phi}) \approx \boldsymbol{I} + [\boldsymbol{\phi}]_\times$ (1차 근사)

예시: $\boldsymbol{\phi} = (0, 0, \pi/2)^\top \to z$축 중심 90° 회전:

$$\mathrm{Exp}\begin{pmatrix} 0 \\ 0 \\ \pi/2 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Logarithm map — 회전을 벡터로

Exp의 역:

$$\mathrm{Log}(\boldsymbol{R}) \;=\; \frac{\theta}{2\sin\theta}\,(\boldsymbol{R}-\boldsymbol{R}^{\top})^{\vee}, \qquad \theta = \arccos\!\left(\frac{\mathrm{tr}(\boldsymbol{R})-1}{2}\right)$$

여기서 $(\cdot)^{\vee}$: 반대칭 행렬에서 3D 벡터 추출 ($[\cdot]_{\times}$의 역)

- 출력: $\boldsymbol{\phi} \in \mathbb{R}^3, \quad \|\boldsymbol{\phi}\| \in [0, \pi]$
- Exp와 Log는 서로 역함수
- $\theta = \pi$에서 불연속 — 실제 시스템에서 피해야 할 조건

Logarithm map — 회전을 벡터로

Exp의 역:

$$\mathrm{Log}(\boldsymbol{R}) \;=\; \frac{\theta}{2\sin\theta}\,(\boldsymbol{R}-\boldsymbol{R}^{\top})^{\vee}, \qquad \theta = \arccos\!\left(\frac{\mathrm{tr}(\boldsymbol{R})-1}{2}\right)$$

여기서 $(\cdot)^{\vee}$: 반대칭 행렬에서 3D 벡터 추출 ($[\cdot]_{\times}$의 역)

- 출력: $\boldsymbol{\phi} \in \mathbb{R}^3, \quad \|\boldsymbol{\phi}\| \in [0, \pi]$
- Exp와 Log는 서로 역함수
- $\theta = \pi$에서 불연속 — 실제 시스템에서 피해야 할 조건

요약

$$\mathbb{R}^3 \xrightarrow{\text{Exp}} \mathrm{SO}(3) \xrightarrow{\text{Log}} \mathbb{R}^3$$

두 공간을 “사다리”처럼 오간다.

Perturbation — 회전의 “덧셈”

SO(3)에서 기준 회전 $\hat{\boldsymbol{R}}$ 주변의 작은 변화를 표현하는 방법: **오른쪽 섭동 (right perturbation):**

$$\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi}) \approx \hat{\boldsymbol{R}}\,(\boldsymbol{I} + [\delta\boldsymbol{\phi}]_\times)$$

Perturbation — 회전의 "덧셈"

SO(3)에서 기준 회전 $\hat{\boldsymbol{R}}$ 주변의 작은 변화를 표현하는 방법: **오른쪽 섭동 (right perturbation):**

$$\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi}) \approx \hat{\boldsymbol{R}}\,(\boldsymbol{I} + [\delta\boldsymbol{\phi}]_\times)$$

- $\delta\boldsymbol{\phi} \in \mathbb{R}^3$ — **제약 없는 벡터**, KF로 추정 가능
- $\delta\boldsymbol{\phi} \to 0$이면 $\boldsymbol{R} \to \hat{\boldsymbol{R}}$ — 명목 회전 주변을 기술
- ESKF에서 rotation error state $\delta\boldsymbol{\phi}$의 의미가 바로 이것

Perturbation — 회전의 "덧셈"

SO(3)에서 기준 회전 $\hat{\boldsymbol{R}}$ 주변의 작은 변화를 표현하는 방법: **오른쪽 섭동 (right perturbation):**

$$\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi}) \approx \hat{\boldsymbol{R}}\,(\boldsymbol{I} + [\delta\boldsymbol{\phi}]_\times)$$

- $\delta\boldsymbol{\phi} \in \mathbb{R}^3$ — **제약 없는 벡터**, KF로 추정 가능
- $\delta\boldsymbol{\phi} \to 0$이면 $\boldsymbol{R} \to \hat{\boldsymbol{R}}$ — 명목 회전 주변을 기술
- ESKF에서 rotation error state $\delta\boldsymbol{\phi}$의 의미가 바로 이것

이게 핵심
쿼터니언은 $|\mathbf{q}| = 1$ 제약 때문에 KF로 직접 갱신할 수 없다.
하지만 $\delta\boldsymbol{\phi} \in \mathbb{R}^3$은 제약이 없다 $\to$ KF로 갱신 가능!
갱신 후: $\hat{\boldsymbol{R}} \leftarrow \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi})$으로 inject, $\delta\boldsymbol{\phi} \leftarrow 0$.

왜 $\partial(\boldsymbol{R}\boldsymbol{v})/\partial\delta\boldsymbol{\phi}$ 가 필요한가? — 동기

ESKF가 추정하는 것은 **error state** $\delta\boldsymbol{\phi}$. 그런데 측정/운동 모델에서 회전은 항상 **"$\boldsymbol{R}$이 어떤 벡터 $\boldsymbol{v}$에 곱해지는"** 형태로 등장한다.

- 가속도계 모델: $\boldsymbol{a}_w = \boldsymbol{R}\,\boldsymbol{a}_b + \boldsymbol{g}$
- 자기장 모델: $\boldsymbol{m}_w = \boldsymbol{R}\,\boldsymbol{m}_b$
- LiDAR 점 변환: $\boldsymbol{p}_w = \boldsymbol{R}\,\boldsymbol{p}_b + \boldsymbol{t}$

왜 $\partial(\boldsymbol{R}\boldsymbol{v})/\partial\delta\boldsymbol{\phi}$ 가 필요한가? — 동기

ESKF가 추정하는 것은 **error state** $\delta\boldsymbol{\phi}$. 그런데 측정/운동 모델에서 회전은 항상 **"$\boldsymbol{R}$이 어떤 벡터 $\boldsymbol{v}$에 곱해지는"** 형태로 등장한다.

- 가속도계 모델: $\boldsymbol{a}_w = \boldsymbol{R}\,\boldsymbol{a}_b + \boldsymbol{g}$
- 자기장 모델: $\boldsymbol{m}_w = \boldsymbol{R}\,\boldsymbol{m}_b$
- LiDAR 점 변환: $\boldsymbol{p}_w = \boldsymbol{R}\,\boldsymbol{p}_b + \boldsymbol{t}$

따라서 ESKF의 **F matrix**(상태 전이)와 **H matrix**(측정 Jacobian)를 유도하려면 반드시 다음 미분이 필요하다:

$$\frac{\partial\,(\boldsymbol{R}\boldsymbol{v})}{\partial\,\delta\boldsymbol{\phi}}$$

("회전이 $\delta\boldsymbol{\phi}$만큼 흔들리면 $\boldsymbol{R}\boldsymbol{v}$는 얼마나 변하나?")

왜 $\partial(\boldsymbol{R}\boldsymbol{v})/\partial\delta\boldsymbol{\phi}$ 가 필요한가? — 동기

ESKF가 추정하는 것은 **error state** $\delta\boldsymbol{\phi}$. 그런데 측정/운동 모델에서 회전은 항상 **"$\boldsymbol{R}$이 어떤 벡터 $\boldsymbol{v}$에 곱해지는"** 형태로 등장한다.

- 가속도계 모델: $\boldsymbol{a}_w = \boldsymbol{R}\,\boldsymbol{a}_b + \boldsymbol{g}$
- 자기장 모델: $\boldsymbol{m}_w = \boldsymbol{R}\,\boldsymbol{m}_b$
- LiDAR 점 변환: $\boldsymbol{p}_w = \boldsymbol{R}\,\boldsymbol{p}_b + \boldsymbol{t}$

따라서 ESKF의 **F matrix**(상태 전이)와 **H matrix**(측정 Jacobian)를 유도하려면 반드시 다음 미분이 필요하다:

$$\frac{\partial\,(\boldsymbol{R}\boldsymbol{v})}{\partial\,\delta\boldsymbol{\phi}}$$

("회전이 $\delta\boldsymbol{\phi}$만큼 흔들리면 $\boldsymbol{R}\boldsymbol{v}$는 얼마나 변하나?")

한 줄 정리
ESKF에서 회전이 들어간 모든 Jacobian은 이 한 미분으로 환원된다.
그래서 다음 슬라이드에서 그 결과 $-\hat{\boldsymbol{R}}[\boldsymbol{v}]_\times$ 를 재료처럼 미리 깔아둔다.

섭동의 Jacobian — ESKF에서 쓰이는 미분

$\delta\boldsymbol{\phi} = 0$ 근방에서 $\boldsymbol{R}(\delta\boldsymbol{\phi}) = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi})$의 Jacobian:

$$\left.\frac{\partial\,(\boldsymbol{R}\boldsymbol{v})}{\partial\,\delta\boldsymbol{\phi}}\right|_{\delta\boldsymbol{\phi}=0} \;=\; -\hat{\boldsymbol{R}}\,[\boldsymbol{v}]_{\times} \;=\; -\left[\hat{\boldsymbol{R}}\boldsymbol{v}\right]_{\times}$$

($\boldsymbol{v}$: 임의의 3D 벡터)

- 이 Jacobian이 ESKF F matrix의 off-diagonal 블록을 만든다 (4회차에서 구체적으로)
- 핵심: $\delta\boldsymbol{\phi} \approx 0$일 때 이 근사가 유효 $\rightarrow$ ESKF는 구조적으로 이를 보장

자주 쓰이는 항등식:

$$\boldsymbol{R}\,[\boldsymbol{a}]_{\times}\,\boldsymbol{R}^{\top} = [\boldsymbol{R}\boldsymbol{a}]_{\times} \qquad \text{(회전 후 외적)}$$

쿼터니언 = SO(3) 구현 도구

실제 코드에서는 회전을 쿼터니언으로 저장하고 계산한다. 대응:

$$\mathrm{Exp}(\boldsymbol{\phi}) \;\leftrightarrow\; \mathbf{q}_{\boldsymbol{\phi}} = \left(\cos\frac{\|\boldsymbol{\phi}\|}{2},\; \frac{\boldsymbol{\phi}}{\|\boldsymbol{\phi}\|}\sin\frac{\|\boldsymbol{\phi}\|}{2}\right)$$

오른쪽 섭동의 쿼터니언 버전:

$$\hat{\mathbf{q}}_{\text{new}} \;=\; \hat{\mathbf{q}} \otimes \mathbf{q}_{\boldsymbol{\delta}}, \qquad \mathbf{q}_{\boldsymbol{\delta}} \approx \left(1,\; \frac{\boldsymbol{\delta\phi}}{2}\right) \quad (\boldsymbol{\delta\phi} \text{ 작을 때})$$

쿼터니언 = SO(3) 구현 도구

실제 코드에서는 회전을 쿼터니언으로 저장하고 계산한다. 대응:

$$\mathrm{Exp}(\boldsymbol{\phi}) \;\leftrightarrow\; \mathbf{q}_{\boldsymbol{\phi}} = \left(\cos\frac{\|\boldsymbol{\phi}\|}{2},\; \frac{\boldsymbol{\phi}}{\|\boldsymbol{\phi}\|}\sin\frac{\|\boldsymbol{\phi}\|}{2}\right)$$

오른쪽 섭동의 쿼터니언 버전:

$$\hat{\mathbf{q}}_{\text{new}} \;=\; \hat{\mathbf{q}} \otimes \mathbf{q}_{\delta}, \qquad \mathbf{q}_{\delta} \approx \left(1,\; \frac{\delta\boldsymbol{\phi}}{2}\right) \quad (\delta\boldsymbol{\phi} \text{ 작을 때})$$

실무 요약

- **명목 회전**: 쿼터니언 $\hat{\mathbf{q}}$로 저장·전파
- **오차 회전**: $\delta\boldsymbol{\phi} \in \mathbb{R}^3$으로 KF가 추정
- **inject**: $\hat{\mathbf{q}} \leftarrow \hat{\mathbf{q}} \otimes (1, \delta\boldsymbol{\phi}/2)$, $\delta\boldsymbol{\phi} \leftarrow 0$

OpenVINS의 skew-symmetric 구현

ov_core/src/utils/quat_ops.h (github.com/rpng/open_vins):

```
// [phi]_x : skew-symmetric matrix of a 3-vector
inline Eigen::Matrix<double,3,3>
skew_x(const Eigen::Matrix<double,3,1> &w) {
  Eigen::Matrix<double,3,3> w_x;
  w_x <<     0, -w(2),  w(1),
          w(2),     0, -w(0),
         -w(1),  w(0),     0;
  return w_x;
}
```

- $[\phi]_\times \boldsymbol{v} = \boldsymbol{\phi} \times \boldsymbol{v}$ — 외적과 수학적으로 동일
- ESKF F matrix의 모든 off-diagonal 블록이 이 함수로 구성됨
- 4회차에서 직접 F matrix를 채울 때 다시 만난다

State vector 정의 — 이 강의 전체에서 쓰이는 표기

ESKF의 상태는 **명목(nominal)** + **오차(error)**로 분리:

명목	오차	물리량	차원
$\hat{\boldsymbol{p}} \in \mathbb{R}^3$	$\delta\boldsymbol{p} \in \mathbb{R}^3$	위치 (world)	3
$\hat{\boldsymbol{v}} \in \mathbb{R}^3$	$\delta\boldsymbol{v} \in \mathbb{R}^3$	속도 (world)	3
$\hat{\mathbf{q}} \in S^3$	$\delta\boldsymbol{\phi} \in \mathbb{R}^3$	회전 (body→world)	3
$\hat{\boldsymbol{b}}_g \in \mathbb{R}^3$	$\delta\boldsymbol{b}_g \in \mathbb{R}^3$	Gyro bias	3
$\hat{\boldsymbol{b}}_a \in \mathbb{R}^3$	$\delta\boldsymbol{b}_a \in \mathbb{R}^3$	Accel bias	3
(저장: 16D)	**KF 차원: 15D**		

회전 오차만 $\delta\boldsymbol{\phi} \in \mathbb{R}^3$ (tangent space) — 모든 오차 성분이 제약 없는 벡터

True state = Nominal $\oplus$ Error

각 성분의 합성 정의:

$$\boldsymbol{p} = \hat{\boldsymbol{p}} + \delta\boldsymbol{p}, \qquad \boldsymbol{v} = \hat{\boldsymbol{v}} + \delta\boldsymbol{v}$$

$$\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi}) \quad (\text{또는 } \hat{\mathbf{q}} \otimes \mathbf{q}_{\delta})$$

$$\boldsymbol{b}_g = \hat{\boldsymbol{b}}_g + \delta\boldsymbol{b}_g, \qquad \boldsymbol{b}_a = \hat{\boldsymbol{b}}_a + \delta\boldsymbol{b}_a$$

True state = Nominal $\oplus$ Error

각 성분의 합성 정의:

$$\boldsymbol{p} = \hat{\boldsymbol{p}} + \delta\boldsymbol{p}, \qquad \boldsymbol{v} = \hat{\boldsymbol{v}} + \delta\boldsymbol{v}$$

$$\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi}) \quad (\text{또는 } \hat{\mathbf{q}} \otimes \mathbf{q}_{\delta})$$

$$\boldsymbol{b}_g = \hat{\boldsymbol{b}}_g + \delta\boldsymbol{b}_g, \qquad \boldsymbol{b}_a = \hat{\boldsymbol{b}}_a + \delta\boldsymbol{b}_a$$

전체 오차 상태

$$\delta\boldsymbol{x} = (\delta\boldsymbol{p},\ \delta\boldsymbol{v},\ \delta\boldsymbol{\phi},\ \delta\boldsymbol{b}_g,\ \delta\boldsymbol{b}_a)^{\top} \in \mathbb{R}^{15}$$

모두 제약 없음 $\rightarrow$ 선형 KF가 $\delta\boldsymbol{x}$를 추정한다

왜 쿼터니언 부호 문제가 ESKF에서 사라지나

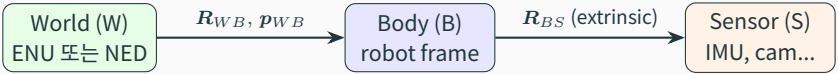
일반 EKF (쿼터니언)

- state에 $\mathbf{q} \in \mathbb{R}^4$ 포함
- $\hat{\mathbf{q}} + K\nu$로 갱신
- $|\mathbf{q}| = 1$ 위반 가능
- $\mathbf{q}$와 $-\mathbf{q}$ 부호 충돌
  $\rightarrow$ 평균이 0 벡터
- renormalize해도 P 방향 오류

ESKF (SO(3) 섭동)

- error state: $\delta\phi \in \mathbb{R}^3$
- KF는 $\delta\phi$만 추정
- $\hat{\mathbf{q}}$: 비선형 방정식으로 전파
- inject: $\hat{\mathbf{q}} \leftarrow \hat{\mathbf{q}} \otimes (1, \delta\phi/2)$
  $\rightarrow$ 항상 단위 구 위
- **부호 문제 구조적 소멸**

좌표 프레임 — World, Body, Sensor



- **World**: 절대 좌표 — 위치·속도가 이 프레임에 표현됨
- **Body**: 로봇 본체 — 각속도·가속도가 이 프레임에 측정됨
- **Sensor**: IMU/카메라 — 캘리브레이션 후 Body로 변환
- ESKF의 $\hat{\mathbf{q}}$ (또는 $\hat{R}$): Body $\rightarrow$ World 변환

비가환성 — 회전은 벡터가 아니다

$$\boldsymbol{R}_1\boldsymbol{R}_2 \neq \boldsymbol{R}_2\boldsymbol{R}_1 \quad \text{(일반적으로)}$$

비가환성 — 회전은 벡터가 아니다

$$\boldsymbol{R}_1\boldsymbol{R}_2 \neq \boldsymbol{R}_2\boldsymbol{R}_1 \quad \text{(일반적으로)}$$

직접 실험:

1. x축 90° $\rightarrow$ z축 90°
2. z축 90° $\rightarrow$ x축 90°

결과가 다르다.

왜 중요한가:

- 회전을 벡터처럼 더하면 틀림
- 회전의 평균도 단순 산술 평균 아님
- ESKF가 $\delta\boldsymbol{\phi}$를 별도로 추적하는 이유

이번 회차 핵심 다섯 개

1. SO(3)는 $\mathbb{R}^9$ 안의 3D 매니폴드 — Lie group
2. $\mathfrak{so}(3) \cong \mathbb{R}^3$ — tangent space에서 작은 회전은 3D 벡터 $\boldsymbol{\phi}$
3. Exp: $\mathbb{R}^3 \to$ SO(3) (Rodrigues), Log: SO$(3) \to \mathbb{R}^3$
4. 오른쪽 섭동: $\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi})$ — KF가 $\delta\boldsymbol{\phi} \in \mathbb{R}^3$를 추정
5. 이 구조 덕분에 ESKF에서 쿼터니언 부호 문제가 구조적으로 소멸

Gaussian + 선형 모델 $\Rightarrow$ 5줄 알고리즘

- Bayes filter + 가우시안 + 선형 모델 = KF
- Predict: $\boldsymbol{P}^{-} = \boldsymbol{F}\boldsymbol{P}^{+}\boldsymbol{F}^{\top} + \boldsymbol{Q}$
- Update: Kalman gain, innovation, **Joseph form (표준)**
- 이 5줄이 ESKF에서 error state $\delta\boldsymbol{x}$에 그대로 적용된다

미리 생각해볼 것
$\boldsymbol{P}^{+} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}^{-}$와
$\boldsymbol{P}^{+} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}^{-}(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})^{\top} + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^{\top}$
중 어느 것이 더 안전하고, 왜인가?

SO(3) Jacobians

— Lie 미분 (Lie Derivatives) —

"Lie 군 위에서 미분하려면 접선공간(tangent space)을 통해야 한다."

참고문헌: Solà (2018) A micro Lie theory for state estimation in robotics
arXiv:1812.01537

왜 SO(3) Jacobian이 필요한가? (동기)

핵심 문제
점 $\boldsymbol{p}' = \boldsymbol{R}\boldsymbol{p}$에서 $\boldsymbol{R}$**에 대한 Jacobian**을 어떻게 구하는가?

$$\frac{\partial(\boldsymbol{R}\boldsymbol{p})}{\partial \boldsymbol{R}} = ?$$

$\boldsymbol{R} \in \mathrm{SO}(3)$는 행렬이지만 **일반 벡터공간이 아님** $\Rightarrow$ 일반 chain rule 불가!

왜 SO(3) Jacobian이 필요한가? (동기)

핵심 문제
점 $\boldsymbol{p}' = \boldsymbol{R}\boldsymbol{p}$에서 $\boldsymbol{R}$**에 대한 Jacobian**을 어떻게 구하는가?

$$\frac{\partial(\boldsymbol{R}\boldsymbol{p})}{\partial \boldsymbol{R}} = ?$$

$\boldsymbol{R} \in \mathrm{SO}(3)$는 행렬이지만 **일반 벡터공간이 아님** $\Rightarrow$ 일반 chain rule 불가!

SO(3) Jacobian이 필요한 곳

- **IMU preintegration**: 회전 오차 공분산 전파
- **ESKF**: so(3) $\leftrightarrow$ SO(3) 공분산 변환
- **Factor graph**: 회전 residual의 Jacobian
- **LiDAR odometry**: pose Jacobian (SE(3))

이 파트의 목표

- Right/Left Jacobian J_r, J_l 정의
- 소각 근사 (practical use)
- BCH 공식과 perturbation 전파
- **Forster 2017 TRO**에서의 핵심 역할

Forster et al. (2017) TRO “On-Manifold Preintegration” – 이 파트의 핵심 응용

섭동(Perturbation) 공식 복습

오른쪽 섭동 (Right Perturbation)
작은 $\boldsymbol{\epsilon} \in \mathbb{R}^3$에 대해:

$$\boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{\epsilon}) \approx \boldsymbol{R}\,(I + [\boldsymbol{\epsilon}]_\times) = \boldsymbol{R} + \boldsymbol{R}\,[\boldsymbol{\epsilon}]_\times$$

섭동(Perturbation) 공식 복습

오른쪽 섭동 (Right Perturbation)

작은 $\boldsymbol{\epsilon} \in \mathbb{R}^3$에 대해:

$$\boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{\epsilon}) \approx \boldsymbol{R}\,(I + [\boldsymbol{\epsilon}]_\times) = \boldsymbol{R} + \boldsymbol{R}\,[\boldsymbol{\epsilon}]_\times$$

Right Perturbation Derivative (방향 미분)

$$\left.\frac{d}{d\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} f(\boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{\epsilon})) \;=\; \lim_{\boldsymbol{\epsilon}\to\mathbf{0}} \frac{f(\boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{\epsilon})) - f(\boldsymbol{R})}{\boldsymbol{\epsilon}}$$

이를 **right perturbation derivative** 또는 **right Jacobian**이라 부른다.

섭동(Perturbation) 공식 복습

오른쪽 섭동 (Right Perturbation)
작은 $\boldsymbol{\epsilon} \in \mathbb{R}^3$에 대해:

$$\boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{\epsilon}) \approx \boldsymbol{R}\,(I + [\boldsymbol{\epsilon}]_\times) = \boldsymbol{R} + \boldsymbol{R}\,[\boldsymbol{\epsilon}]_\times$$

Right Perturbation Derivative (방향 미분)

$$\left.\frac{d}{d\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} f(\boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{\epsilon})) \;=\; \lim_{\boldsymbol{\epsilon}\to\mathbf{0}} \frac{f(\boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{\epsilon})) - f(\boldsymbol{R})}{\boldsymbol{\epsilon}}$$

이를 **right perturbation derivative** 또는 **right Jacobian**이라 부른다.

예시: $f(\boldsymbol{R}) = \boldsymbol{R}\boldsymbol{p}$

$$\begin{aligned}\left.\frac{d}{d\boldsymbol{\epsilon}}\right|_{\mathbf{0}} \boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{\epsilon})\,\boldsymbol{p} &\approx \frac{d}{d\boldsymbol{\epsilon}}\boldsymbol{R}(I + [\boldsymbol{\epsilon}]_\times)\boldsymbol{p} \\ &= \boldsymbol{R}\cdot\frac{d\,[\boldsymbol{\epsilon}]_\times\,\boldsymbol{p}}{d\boldsymbol{\epsilon}} \\ &= -\boldsymbol{R}\,[\boldsymbol{p}]_\times\end{aligned}$$

직관
SO(3) 위의 함수 f를 미분하려면:

$\boldsymbol{R}$에 **오른쪽으로** 작은 Lie algebra 원소 $\mathrm{Exp}(\epsilon)$를 곱한 뒤 극한을 취한다.

$\Rightarrow$ 이것이 **SO(3)에서의 미분**의 정의

Right Jacobian $J_r(\phi)$ — 정의

문제 설정
$\phi \in \mathbb{R}^3$에 작은 변분 $\delta\phi$를 더할 때, $\mathrm{Exp}(\phi + \delta\phi)$를 어떻게 표현?

Right Jacobian의 역할 (핵심 공식)

$$\mathrm{Exp}(\phi + \delta\phi) \approx \mathrm{Exp}(\phi)\,\mathrm{Exp}(J_r(\phi)\,\delta\phi)$$

단, $\delta\phi$가 충분히 작을 때 (1차 근사)

Right Jacobian $J_r(\phi)$ — 정의

문제 설정
$\phi \in \mathbb{R}^3$에 작은 변분 $\delta\phi$를 더할 때, $\mathrm{Exp}(\phi + \delta\phi)$를 어떻게 표현?

Right Jacobian의 역할 (핵심 공식)

$$\boxed{\mathrm{Exp}(\phi + \delta\phi) \approx \mathrm{Exp}(\phi)\,\mathrm{Exp}(J_r(\phi)\,\delta\phi)}$$

단, $\delta\phi$가 충분히 작을 때 (1차 근사)

적분 정의 (정확)

$$J_r(\phi) = \int_0^1 \mathrm{Exp}(-(1-s)\,\phi)\; ds \;\in\; \mathbb{R}^{3\times 3}$$

Right Jacobian $J_r(\phi)$ — 정의

문제 설정
$\phi \in \mathbb{R}^3$에 작은 변분 $\delta\phi$를 더할 때, $\mathrm{Exp}(\phi + \delta\phi)$를 어떻게 표현?

Right Jacobian의 역할 (핵심 공식)

$$\boxed{\mathrm{Exp}(\phi + \delta\phi) \approx \mathrm{Exp}(\phi)\,\mathrm{Exp}(J_r(\phi)\,\delta\phi)}$$

단, $\delta\phi$가 충분히 작을 때 (1차 근사)

적분 정의 (정확)

$$J_r(\phi) = \int_0^1 \mathrm{Exp}(-(1-s)\,\phi)\; ds \;\in\; \mathbb{R}^{3\times 3}$$

폐쇄형 (Closed-form)
$\phi = \|\phi\|$, $\hat{\phi} = \phi/\phi$로 쓰면:

$$J_r(\phi) = I - \frac{1-\cos\phi}{\phi^2}\,[\phi]_\times + \frac{\phi - \sin\phi}{\phi^3}\,[\phi]_\times^2$$

(Rodrigues 공식과 유사한 구조)

Right Jacobian $J_r(\phi)$ — 정의

문제 설정
$\boldsymbol{\phi} \in \mathbb{R}^3$에 작은 변분 $\delta\boldsymbol{\phi}$를 더할 때, $\mathrm{Exp}(\boldsymbol{\phi} + \delta\boldsymbol{\phi})$를 어떻게 표현?

Right Jacobian의 역할 (핵심 공식)

$$\boxed{\mathrm{Exp}(\boldsymbol{\phi} + \delta\boldsymbol{\phi}) \approx \mathrm{Exp}(\boldsymbol{\phi})\,\mathrm{Exp}(J_r(\boldsymbol{\phi})\,\delta\boldsymbol{\phi})}$$

단, $\delta\boldsymbol{\phi}$가 충분히 작을 때 (1차 근사)

적분 정의 (정확)

$$J_r(\boldsymbol{\phi}) = \int_0^1 \mathrm{Exp}(-(1-s)\,\boldsymbol{\phi})\; ds \;\in\; \mathbb{R}^{3\times 3}$$

폐쇄형 (Closed-form)
$\phi = \|\boldsymbol{\phi}\|$, $\hat{\boldsymbol{\phi}} = \boldsymbol{\phi}/\phi$로 쓰면:

$$J_r(\boldsymbol{\phi}) = I - \frac{1-\cos\phi}{\phi^2}\,[\boldsymbol{\phi}]_\times + \frac{\phi - \sin\phi}{\phi^3}\,[\boldsymbol{\phi}]_\times^2$$

(Rodrigues 공식과 유사한 구조)

소각 극한
$\boldsymbol{\phi} \to \mathbf{0}$ 이면 $J_r(\boldsymbol{\phi}) \to I_{3\times 3}$

Left Jacobian $J_l(\phi)$ — 정의

왼쪽 섭동 기반 정의
Left Jacobian J_l은 **왼쪽 섭동**에서 자연스럽게 등장:

$$\mathrm{Exp}(\phi + \delta\phi) \approx \mathrm{Exp}\left(J_l^{-1}(\phi)\,\delta\phi\right)\ \mathrm{Exp}(\phi)$$

Left Jacobian $J_l(\phi)$ — 정의

왼쪽 섭동 기반 정의
Left Jacobian J_l은 **왼쪽 섭동**에서 자연스럽게 등장:

$$\mathrm{Exp}(\phi + \delta\phi) \approx \mathrm{Exp}\left(J_l^{-1}(\phi)\,\delta\phi\right)\,\mathrm{Exp}(\phi)$$

폐쇄형 (Closed-form)

$$J_l(\phi) = I + \frac{1-\cos\phi}{\phi^2}\,[\phi]_\times + \frac{\phi-\sin\phi}{\phi^3}\,[\phi]_\times^2$$

J_r과 부호 차이(skew 항)에 주의!

Left Jacobian $J_l(\phi)$ — 정의

왼쪽 섭동 기반 정의
Left Jacobian J_l은 **왼쪽 섭동**에서 자연스럽게 등장:

$$\mathrm{Exp}(\boldsymbol{\phi} + \delta\boldsymbol{\phi}) \approx \mathrm{Exp}\left(J_l^{-1}(\boldsymbol{\phi})\,\delta\boldsymbol{\phi}\right)\,\mathrm{Exp}(\boldsymbol{\phi})$$

폐쇄형 (Closed-form)

$$J_l(\boldsymbol{\phi}) = I + \frac{1-\cos\phi}{\phi^2}\,[\boldsymbol{\phi}]_\times + \frac{\phi-\sin\phi}{\phi^3}\,[\boldsymbol{\phi}]_\times^2$$

J_r과 부호 차이(skew 항)에 주의!

관계식 1: 반전

$$J_l(\boldsymbol{\phi}) = J_r(-\boldsymbol{\phi})$$

$\boldsymbol{\phi}$ 방향을 뒤집으면 Left ↔ Right 교환

관계식 2: Adjoint

$$J_l(\boldsymbol{\phi}) = \boldsymbol{R}\,J_r(\boldsymbol{\phi})$$

단, $\boldsymbol{R} = \mathrm{Exp}(\boldsymbol{\phi})$

(Adjoint representation으로부터 유도됨; SO3J-7 참고)

실용 선택 기준
ESKF의 **right perturbation** 관행에서는 J_r이 주로 사용된다.
Barfoot (2017) 교재는 Left Jacobian 중심으로 전개한다.

Right Jacobian 역행렬 $J_r^{-1}(\boldsymbol{\phi})$

폐쇄형

$$J_r^{-1}(\boldsymbol{\phi}) = \frac{\phi}{2}\cot\frac{\phi}{2}\,I + \left(1 - \frac{\phi}{2}\cot\frac{\phi}{2}\right)\hat{\boldsymbol{\phi}}\hat{\boldsymbol{\phi}}^{\top} + \frac{\phi}{2}\,\left[\hat{\boldsymbol{\phi}}\right]_{\times}$$

단, $\phi = \|\boldsymbol{\phi}\|$, $\hat{\boldsymbol{\phi}} = \boldsymbol{\phi}/\phi$

Right Jacobian 역행렬 $J_r^{-1}(\boldsymbol{\phi})$

폐쇄형

$$J_r^{-1}(\boldsymbol{\phi}) = \frac{\phi}{2}\cot\frac{\phi}{2}\,I + \left(1 - \frac{\phi}{2}\cot\frac{\phi}{2}\right)\hat{\boldsymbol{\phi}}\hat{\boldsymbol{\phi}}^\top + \frac{\phi}{2}\left[\hat{\boldsymbol{\phi}}\right]_\times$$

단, $\phi = \|\boldsymbol{\phi}\|$, $\hat{\boldsymbol{\phi}} = \boldsymbol{\phi}/\phi$

동치 표현 (급수 이용)

$$J_r^{-1}(\boldsymbol{\phi}) = \sum_{n=0}^{\infty} \frac{B_n}{n!}\,[\boldsymbol{\phi}]_\times^n$$

B_n: Bernoulli 수 ($B_0 = 1, B_1 = \frac{1}{2}, B_2 = \frac{1}{6}, \ldots$)

1차 Taylor 근사 (소각)

$$J_r^{-1}(\boldsymbol{\phi}) \approx I + \frac{1}{2}\,[\boldsymbol{\phi}]_\times$$

오차: $O(\|\boldsymbol{\phi}\|^2)$

Right Jacobian 역행렬 $J_r^{-1}(\boldsymbol{\phi})$

폐쇄형

$$J_r^{-1}(\boldsymbol{\phi}) = \frac{\phi}{2}\cot\frac{\phi}{2}\,I + \left(1 - \frac{\phi}{2}\cot\frac{\phi}{2}\right)\hat{\boldsymbol{\phi}}\hat{\boldsymbol{\phi}}^\top + \frac{\phi}{2}\left[\hat{\boldsymbol{\phi}}\right]_\times$$

단, $\phi = \|\boldsymbol{\phi}\|$, $\hat{\boldsymbol{\phi}} = \boldsymbol{\phi}/\phi$

동치 표현 (급수 이용)

$$J_r^{-1}(\boldsymbol{\phi}) = \sum_{n=0}^{\infty} \frac{B_n}{n!}\,[\boldsymbol{\phi}]_\times^n$$

B_n: Bernoulli 수 ($B_0 = 1, B_1 = \frac{1}{2}, B_2 = \frac{1}{6}, \ldots$)

1차 Taylor 근사 (소각)

$$J_r^{-1}(\boldsymbol{\phi}) \approx I + \frac{1}{2}\,[\boldsymbol{\phi}]_\times$$

오차: $O(\|\boldsymbol{\phi}\|^2)$

ESKF에서의 실용

noise covariance 변환: IMU gyro noise $\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, Q_d)$가 **so(3)**에서 정의될 때, SO(3) 공간으로 전파되는 공분산은:

$$\Sigma_{\mathrm{SO}(3)} = J_r\, Q_d\, J_r^\top \qquad \Longleftrightarrow \qquad Q_d = J_r^{-1}\,\Sigma_{\mathrm{SO}(3)}\,(J_r^{-1})^\top$$

소각에서 $J_r \approx I$이면 거의 같아지지만, $\|\boldsymbol{\phi}\| \gtrsim 10^\circ$이면 차이 발생

소각 근사 (Small-angle Approximation)

J_r 소각 근사

$$J_r(\phi) \approx I - \frac{1}{2}[\phi]_\times$$

(1차 Taylor, $O(\|\phi\|^2)$ 오차)

J_r^{-1} 소각 근사

$$J_r^{-1}(\phi) \approx I + \frac{1}{2}[\phi]_\times$$

둘을 곱하면: $(I - \frac{1}{2}[\phi]_\times)(I + \frac{1}{2}[\phi]_\times) \approx I$
(2차 이상 무시 시 정확)

ESKF에서의 표준 가정
Error state $\delta\phi$는 **항상 소각**
(nominal 상태 근방에서 동작)

$\Rightarrow$ 소각 근사를 **표준**으로 사용
$\Rightarrow$ $J_r \approx I$까지 단순화하기도 함

소각 근사 (Small-angle Approximation)

J_r 소각 근사

$$J_r(\boldsymbol{\phi}) \approx I - \frac{1}{2}[\boldsymbol{\phi}]_\times$$

(1차 Taylor, $O(\|\boldsymbol{\phi}\|^2)$ 오차)

J_r^{-1} 소각 근사

$$J_r^{-1}(\boldsymbol{\phi}) \approx I + \frac{1}{2}[\boldsymbol{\phi}]_\times$$

둘을 곱하면: $(I - \frac{1}{2}[\boldsymbol{\phi}]_\times)(I + \frac{1}{2}[\boldsymbol{\phi}]_\times) \approx I$
(2차 이상 무시 시 정확)

ESKF에서의 표준 가정
Error state $\delta\boldsymbol{\phi}$는 **항상 소각**
(nominal 상태 근방에서 동작)

$\Rightarrow$ 소각 근사를 **표준**으로 사용
$\Rightarrow$ $J_r \approx I$까지 단순화하기도 함

수치 비교: $\boldsymbol{\phi} = \phi\,\hat{e}_z$ (z축 회전)

ϕ	J_r (11 성분, 정확)	J_r (근사)	오차 (Frobenius)
5°	0.9962	0.9962	$< 0.01\%$
10°	0.9848	0.9848	$< 0.05\%$
30°	0.9093	0.9133	$\sim 0.5\%$
90°	0.6366	0.7854	$\sim 23\%$

(대각 성분 기준 / skew 방향 성분은 부호만 반전)

소각 근사 (Small-angle Approximation)

J_r 소각 근사

$$J_r(\boldsymbol{\phi}) \approx I - \frac{1}{2}[\boldsymbol{\phi}]_\times$$

(1차 Taylor, $O(\|\boldsymbol{\phi}\|^2)$ 오차)

J_r^{-1} 소각 근사

$$J_r^{-1}(\boldsymbol{\phi}) \approx I + \frac{1}{2}[\boldsymbol{\phi}]_\times$$

둘을 곱하면: $(I - \frac{1}{2}[\boldsymbol{\phi}]_\times)(I + \frac{1}{2}[\boldsymbol{\phi}]_\times) \approx I$
(2차 이상 무시 시 정확)

ESKF에서의 표준 가정
Error state $\delta\boldsymbol{\phi}$는 **항상 소각**
(nominal 상태 근방에서 동작)

$\Rightarrow$ 소각 근사를 **표준**으로 사용
$\Rightarrow$ $J_r \approx I$까지 단순화하기도 함

수치 비교: $\boldsymbol{\phi} = \phi\,\hat{e}_z$ (z축 회전)

ϕ	J_r (11 성분, 정확)	J_r (근사)	오차 (Frobenius)
5°	0.9962	0.9962	$< 0.01\%$
10°	0.9848	0.9848	$< 0.05\%$
30°	0.9093	0.9133	$\sim 0.5\%$
90°	0.6366	0.7854	$\sim 23\%$

(대각 성분 기준 / skew 방향 성분은 부호만 반전)

결론
$\|\boldsymbol{\phi}\| \lesssim 10^\circ$ (≈ 0.17 rad): 소각 근사로 충분히 정확
대각 이동이 큰 경우($\gtrsim 30^\circ$): 폐쇄형 J_r 필요

Adjoint Representation $\mathrm{Ad}_{\boldsymbol{R}}$

Conjugation 성질 (핵심)
임의의 $\boldsymbol{R} \in \mathrm{SO}(3)$, $\boldsymbol{\phi} \in \mathbb{R}^3$에 대해:

$$\boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{\phi})\,\boldsymbol{R}^{\top} = \mathrm{Exp}(\boldsymbol{R}\,\boldsymbol{\phi})$$

즉, $\boldsymbol{R}$으로 켤레(conjugation)하면 지수 안의 벡터도 $\boldsymbol{R}$로 회전된다.

Adjoint Representation $\mathrm{Ad}_{\boldsymbol{R}}$

Conjugation 성질 (핵심)
임의의 $\boldsymbol{R} \in \mathrm{SO}(3), \boldsymbol{\phi} \in \mathbb{R}^3$에 대해:

$$\boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{\phi})\,\boldsymbol{R}^\top = \mathrm{Exp}(\boldsymbol{R}\,\boldsymbol{\phi})$$

즉, $\boldsymbol{R}$으로 켤레(conjugation)하면 지수 안의 벡터도 $\boldsymbol{R}$로 회전된다.

Adjoint 정의

$$\mathrm{Ad}_{\boldsymbol{R}} : \mathfrak{so}(3) \to \mathfrak{so}(3), \qquad \mathrm{Ad}_{\boldsymbol{R}}(\boldsymbol{\phi}) = \boldsymbol{R}\,\boldsymbol{\phi}$$

SO(3)의 경우 Adjoint 행렬은 $\boldsymbol{R}$ 자체: $\mathrm{Ad}_{\boldsymbol{R}} = \boldsymbol{R} \in \mathbb{R}^{3\times 3}$

Adjoint Representation $\mathrm{Ad}_{\boldsymbol{R}}$

Conjugation 성질 (핵심)
임의의 $\boldsymbol{R} \in \mathrm{SO}(3), \boldsymbol{\phi} \in \mathbb{R}^3$에 대해:

$$\boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{\phi})\,\boldsymbol{R}^\top = \mathrm{Exp}(\boldsymbol{R}\,\boldsymbol{\phi})$$

즉, $\boldsymbol{R}$으로 켤레(conjugation)하면 지수 안의 벡터도 $\boldsymbol{R}$로 회전된다.

Adjoint 정의

$$\mathrm{Ad}_{\boldsymbol{R}} : \mathfrak{so}(3) \to \mathfrak{so}(3), \qquad \mathrm{Ad}_{\boldsymbol{R}}(\boldsymbol{\phi}) = \boldsymbol{R}\,\boldsymbol{\phi}$$

SO(3)의 경우 Adjoint 행렬은 $\boldsymbol{R}$ 자체: $\mathrm{Ad}_{\boldsymbol{R}} = \boldsymbol{R} \in \mathbb{R}^{3\times 3}$

Left ↔ Right 변환

$$\boldsymbol{R}\,\mathrm{Exp}(\boldsymbol{\phi}) = \mathrm{Exp}(\boldsymbol{R}\boldsymbol{\phi})\,\boldsymbol{R}$$

⇒ 오른쪽 섭동을 왼쪽으로 옮기면 $\boldsymbol{R}$이 적용됨

Jacobian에 적용

$$J_l(\boldsymbol{\phi}) = \boldsymbol{R}\,J_r(\boldsymbol{\phi}), \quad \boldsymbol{R} = \mathrm{Exp}(\boldsymbol{\phi})$$

Left/Right Jacobian의 관계가 Adjoint로 설명됨

SE(3)로의 확장 (참고)
SE(3)에서는: $\mathrm{Ad}_{\boldsymbol{T}} = \begin{pmatrix} \boldsymbol{R} & [\boldsymbol{t}]_\times \boldsymbol{R} \\ \boldsymbol{0} & \boldsymbol{R} \end{pmatrix} \in \mathbb{R}^{6\times 6}$ (L2 마지막 슬라이드에서 간략 소개)

BCH (Baker-Campbell-Hausdorff) 공식

정확한 BCH 공식 (실용적이지 않음)

$$\text{Log}(\text{Exp}(\boldsymbol{a})\,\text{Exp}(\boldsymbol{b})) = \boldsymbol{a} + \boldsymbol{b} + \tfrac{1}{2}[\boldsymbol{a}, \boldsymbol{b}] + \tfrac{1}{12}\left([\boldsymbol{a}, [\boldsymbol{a}, \boldsymbol{b}]] + [\boldsymbol{b}, [\boldsymbol{b}, \boldsymbol{a}]]\right) + \cdots$$

$[\boldsymbol{a}, \boldsymbol{b}] = [\boldsymbol{a}]_\times \boldsymbol{b} - [\boldsymbol{b}]_\times \boldsymbol{a}$: Lie bracket (SO(3)에서 cross product)

BCH (Baker-Campbell-Hausdorff) 공식

정확한 BCH 공식 (실용적이지 않음)

$$\text{Log}(\text{Exp}(\boldsymbol{a})\,\text{Exp}(\boldsymbol{b})) = \boldsymbol{a} + \boldsymbol{b} + \tfrac{1}{2}[\boldsymbol{a}, \boldsymbol{b}] + \tfrac{1}{12}\big([\boldsymbol{a}, [\boldsymbol{a}, \boldsymbol{b}]] + [\boldsymbol{b}, [\boldsymbol{b}, \boldsymbol{a}]]\big) + \cdots$$

$[\boldsymbol{a}, \boldsymbol{b}] = [\boldsymbol{a}]_\times \boldsymbol{b} - [\boldsymbol{b}]_\times \boldsymbol{a}$: Lie bracket (SO(3)에서 cross product)

1차 BCH 근사 (소각 b, 임의 a)

$$\boxed{\text{Log}(\text{Exp}(\boldsymbol{a})\,\text{Exp}(\boldsymbol{b})) \approx \boldsymbol{a} + J_l^{-1}(\boldsymbol{a})\,\boldsymbol{b}} \quad (\|\boldsymbol{b}\| \text{ 소각})$$

대칭: 소각 $\boldsymbol{a}$, 임의 $\boldsymbol{b}$이면:

$$\text{Log}(\text{Exp}(\boldsymbol{a})\,\text{Exp}(\boldsymbol{b})) \approx \boldsymbol{b} + J_r^{-1}(\boldsymbol{b})\,\boldsymbol{a}$$

BCH (Baker-Campbell-Hausdorff) 공식

정확한 BCH 공식 (실용적이지 않음)

$$\mathrm{Log}(\mathrm{Exp}(\boldsymbol{a})\,\mathrm{Exp}(\boldsymbol{b})) = \boldsymbol{a} + \boldsymbol{b} + \tfrac{1}{2}[\boldsymbol{a},\boldsymbol{b}] + \tfrac{1}{12}\left([\boldsymbol{a},[\boldsymbol{a},\boldsymbol{b}]] + [\boldsymbol{b},[\boldsymbol{b},\boldsymbol{a}]]\right) + \cdots$$

$[\boldsymbol{a},\boldsymbol{b}] = [\boldsymbol{a}]_\times \boldsymbol{b} - [\boldsymbol{b}]_\times \boldsymbol{a}$: Lie bracket (SO(3)에서 cross product)

1차 BCH 근사 (소각 b, 임의 a)

$$\boxed{\mathrm{Log}(\mathrm{Exp}(\boldsymbol{a})\,\mathrm{Exp}(\boldsymbol{b})) \approx \boldsymbol{a} + J_l^{-1}(\boldsymbol{a})\,\boldsymbol{b}} \quad (\|\boldsymbol{b}\| \text{ 소각})$$

대칭: 소각 $\boldsymbol{a}$, 임의 $\boldsymbol{b}$이면:

$$\mathrm{Log}(\mathrm{Exp}(\boldsymbol{a})\,\mathrm{Exp}(\boldsymbol{b})) \approx \boldsymbol{b} + J_r^{-1}(\boldsymbol{b})\,\boldsymbol{a}$$

0차 근사 (둘 다 소각)

$$\mathrm{Log}(\mathrm{Exp}(\boldsymbol{a})\,\mathrm{Exp}(\boldsymbol{b})) \approx \boldsymbol{a} + \boldsymbol{b}$$

Lie algebra가 가환이 되는 극한 – 매우 작은 각도에서만 유효

Preintegration에서의 사용

시간 k에서 $k+1$로 회전 합성:

$$\boldsymbol{R}_{k+1} = \boldsymbol{R}_k\,\mathrm{Exp}(\Delta\boldsymbol{\phi}_k)$$

오차 $\delta\boldsymbol{\phi}$ 전파 시 BCH 1차 근사 사용 $\Rightarrow J_r^{-1}$ 등장

섭동 전파 공식 (핵심 요약)

핵심 공식 — Right Perturbation

$$\boxed{\mathrm{Exp}(\phi + \delta\phi) = \mathrm{Exp}(\phi)\,\mathrm{Exp}(J_r(\phi)\,\delta\phi)}$$

이것은 **정확한** 공식이 아니라 BCH 1차 근사이지만, ESKF에서 표준으로 사용

섭동 전파 공식 (핵심 요약)

핵심 공식 — Right Perturbation

$$\boxed{\mathrm{Exp}(\phi + \delta\phi) = \mathrm{Exp}(\phi)\,\mathrm{Exp}(J_r(\phi)\,\delta\phi)}$$

이것은 **정확한** 공식이 아니라 BCH 1차 근사이지만, ESKF에서 표준으로 사용

Left 버전

$$\mathrm{Exp}(\phi + \delta\phi) = \mathrm{Exp}(J_l(\phi)\,\delta\phi)\ \mathrm{Exp}(\phi)$$

섭동 전파 공식 (핵심 요약)

핵심 공식 — Right Perturbation

$$\boxed{\mathrm{Exp}(\phi + \delta\phi) = \mathrm{Exp}(\phi)\,\mathrm{Exp}(J_r(\phi)\,\delta\phi)}$$

이것은 **정확한** 공식이 아니라 BCH 1차 근사이지만, ESKF에서 표준으로 사용

Left 버전

$$\mathrm{Exp}(\phi + \delta\phi) = \mathrm{Exp}(J_l(\phi)\,\delta\phi)\ \mathrm{Exp}(\phi)$$

물리적 의미
SO(3)에서 작은 perturbation $\delta\phi$를 더하는 것은
ϕ 공간의 "선형 공간"이 아니라
$J_r(\phi)$로 **왜곡된 공간**에서의 이동에 해당

$\Rightarrow$ J_r이 이 "왜곡"을 보정

ESKF update: inject & reset

1. **Inject**: $\hat{\boldsymbol{R}} \leftarrow \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\hat{\boldsymbol{\phi}})$
2. **Reset**: $\delta\hat{\boldsymbol{\phi}} \leftarrow \mathbf{0}$
3. 공분산 reset (Jacobian G):

$$G = I - \frac{1}{2}\left[\delta\hat{\boldsymbol{\phi}}\right]_\times$$

$\Rightarrow$ 이 모든 단계에 perturbation 공식이 관여

섭동 전파 공식 (핵심 요약)

핵심 공식 — Right Perturbation

$$\boxed{\text{Exp}(\boldsymbol{\phi} + \delta\boldsymbol{\phi}) = \text{Exp}(\boldsymbol{\phi})\,\text{Exp}(J_r(\boldsymbol{\phi})\,\delta\boldsymbol{\phi})}$$

이것은 **정확한** 공식이 아니라 BCH 1차 근사이지만, ESKF에서 표준으로 사용

Left 버전

$$\text{Exp}(\boldsymbol{\phi} + \delta\boldsymbol{\phi}) = \text{Exp}(J_l(\boldsymbol{\phi})\,\delta\boldsymbol{\phi})\ \text{Exp}(\boldsymbol{\phi})$$

물리적 의미

SO(3)에서 작은 perturbation $\delta\boldsymbol{\phi}$를 더하는 것은
$\boldsymbol{\phi}$ 공간의 "선형 공간"이 아니라
$J_r(\boldsymbol{\phi})$로 **왜곡된 공간**에서의 이동에 해당

$\Rightarrow J_r$이 이 "왜곡"을 보정

ESKF update: inject & reset

1. **Inject**: $\hat{\boldsymbol{R}} \leftarrow \hat{\boldsymbol{R}}\,\text{Exp}(\delta\hat{\boldsymbol{\phi}})$
2. **Reset**: $\delta\hat{\boldsymbol{\phi}} \leftarrow \mathbf{0}$
3. 공분산 reset (Jacobian G):

$$G = I - \frac{1}{2}\left[\delta\hat{\boldsymbol{\phi}}\right]_\times$$

$\Rightarrow$ 이 모든 단계에 perturbation 공식이 관여

역 방향: Log 유도

$\boldsymbol{R}_1 = \text{Exp}(\boldsymbol{\phi}_1)$, $\boldsymbol{R}_2 = \boldsymbol{R}_1\,\text{Exp}(\delta\boldsymbol{\phi})$이면:

$$\delta\boldsymbol{\phi} = J_r^{-1}(\boldsymbol{\phi}_1)\,\text{Log}(\boldsymbol{R}_1^\top \boldsymbol{R}_2)$$

공분산 변환: so(3) ↔ SO(3)

문제 설정
IMU gyro noise는 **so(3) 접선공간**에서 정의:

$$\boldsymbol{n}_\omega \sim \mathcal{N}(\mathbf{0},\, Q_d) \quad \text{(연속: } \sigma_g^2 I\text{)}$$

이 noise가 **SO(3) 공간의 회전**으로 전파될 때 공분산은?

공분산 변환: so(3) ↔ SO(3)

문제 설정
IMU gyro noise는 **so(3) 접선공간**에서 정의:

$$\boldsymbol{n}_\omega \sim \mathcal{N}(\mathbf{0},\, Q_d) \quad \text{(연속: } \sigma_g^2 I\text{)}$$

이 noise가 **SO(3) 공간의 회전**으로 전파될 때 공분산은?

공분산 전파 공식

$$\boldsymbol{R}_{\text{noisy}} = \text{Exp}(\boldsymbol{\phi} + \boldsymbol{n}) \approx \text{Exp}(\boldsymbol{\phi})\, \text{Exp}(J_r(\boldsymbol{\phi})\, \boldsymbol{n})$$

따라서 SO(3) 공간에서의 유효 noise covariance:

$$\boxed{\Sigma_{\text{SO(3)}} = J_r(\boldsymbol{\phi})\, Q_d\, J_r(\boldsymbol{\phi})^\top}$$

역방향:

$$Q_d = J_r^{-1}(\boldsymbol{\phi})\, \Sigma_{\text{SO(3)}}\, \left(J_r^{-1}(\boldsymbol{\phi})\right)^\top$$

공분산 변환: so(3) ↔ SO(3)

문제 설정
IMU gyro noise는 **so(3) 접선공간**에서 정의:

$$\boldsymbol{n}_\omega \sim \mathcal{N}(\mathbf{0},\, Q_d) \quad \text{(연속: } \sigma_g^2 I\text{)}$$

이 noise가 **SO(3) 공간의 회전**으로 전파될 때 공분산은?

공분산 전파 공식

$$\boldsymbol{R}_{\text{noisy}} = \text{Exp}(\boldsymbol{\phi} + \boldsymbol{n}) \approx \text{Exp}(\boldsymbol{\phi})\, \text{Exp}(J_r(\boldsymbol{\phi})\, \boldsymbol{n})$$

따라서 SO(3) 공간에서의 유효 noise covariance:

$$\boxed{\Sigma_{\text{SO(3)}} = J_r(\boldsymbol{\phi})\, Q_d\, J_r(\boldsymbol{\phi})^\top}$$

역방향:

$$Q_d = J_r^{-1}(\boldsymbol{\phi})\, \Sigma_{\text{SO(3)}}\, \left(J_r^{-1}(\boldsymbol{\phi})\right)^\top$$

소각 근사에서
$\boldsymbol{\phi} \to \mathbf{0}$이면 $J_r \to I$:

수치 예시
$\sigma_g = 0.01$ rad/s, $\Delta t = 0.01$ s,
$\phi = 5^\circ \approx 0.087$ rad 회전 후:

Right Jacobian 도출: 단계별 유도

출발점: Exp 미분
$\boldsymbol{\phi}(t)$ 경로를 따라 $\mathrm{Exp}(\boldsymbol{\phi}(t))$ 미분:

$$\frac{d}{dt}\mathrm{Exp}(\boldsymbol{\phi}(t)) = \mathrm{Exp}(\boldsymbol{\phi}(t))\ \left[J_r(\boldsymbol{\phi}(t))\,\dot{\boldsymbol{\phi}}(t)\right]_\times$$

이것이 J_r의 운동학적 의미 — SO(3)의 body angular velocity와 연결

Right Jacobian 도출: 단계별 유도

출발점: Exp 미분
$\boldsymbol{\phi}(t)$ 경로를 따라 $\mathrm{Exp}(\boldsymbol{\phi}(t))$ 미분:

$$\frac{d}{dt}\mathrm{Exp}(\boldsymbol{\phi}(t)) = \mathrm{Exp}(\boldsymbol{\phi}(t))\ \left[J_r(\boldsymbol{\phi}(t))\,\dot{\boldsymbol{\phi}}(t)\right]_\times$$

이것이 J_r의 운동학적 의미 — SO(3)의 body angular velocity와 연결

적분 유도 ($s \in [0,1]$ 매개변수화)
$\boldsymbol{\phi}(s) = s\,\boldsymbol{\phi}$로 놓으면, 적분을 통해:

$$J_r(\boldsymbol{\phi}) = \int_0^1 \mathrm{Exp}(-(1-s)\,\boldsymbol{\phi})\ ds$$

피적분함수는 Rodrigues 공식으로 계산 가능 (3×3 행렬)

Right Jacobian 도출: 단계별 유도

출발점: Exp 미분
$\boldsymbol{\phi}(t)$ 경로를 따라 $\mathrm{Exp}(\boldsymbol{\phi}(t))$ 미분:

$$\frac{d}{dt}\mathrm{Exp}(\boldsymbol{\phi}(t)) = \mathrm{Exp}(\boldsymbol{\phi}(t)) \left[J_r(\boldsymbol{\phi}(t))\, \dot{\boldsymbol{\phi}}(t) \right]_\times$$

이것이 J_r의 운동학적 의미 — SO(3)의 body angular velocity와 연결

적분 유도 ($s \in [0,1]$ 매개변수화)
$\boldsymbol{\phi}(s) = s\,\boldsymbol{\phi}$로 놓으면, 적분을 통해:

$$J_r(\boldsymbol{\phi}) = \int_0^1 \mathrm{Exp}(-(1-s)\,\boldsymbol{\phi})\; ds$$

피적분함수는 Rodrigues 공식으로 계산 가능 (3×3 행렬)

계산 과정 (핵심 단계만)
$\mathrm{Exp}(-\alpha\boldsymbol{\phi}) = I - \sin(\alpha\phi)\,\hat{\boldsymbol{\phi}}_\times + (1-\cos(\alpha\phi))\,\hat{\boldsymbol{\phi}}_\times^2$ 이므로:

$$\begin{aligned} J_r &= \int_0^1 \left[I - \sin((1-s)\phi) \left[\hat{\boldsymbol{\phi}}\right]_\times + (1-\cos((1-s)\phi)) \left[\hat{\boldsymbol{\phi}}\right]_\times^2 \right] ds \\ &= I - \frac{1-\cos\phi}{\phi} \left[\hat{\boldsymbol{\phi}}\right]_\times + \frac{\phi - \sin\phi}{\phi} \left[\hat{\boldsymbol{\phi}}\right]_\times^2 \end{aligned}$$

$\hat{\boldsymbol{\phi}}_\times = \left[\hat{\boldsymbol{\phi}}\right]_\times = [\boldsymbol{\phi}]_\times / \phi$ 대입하면 폐쇄형 획득

실습 코드: Right Jacobian 구현 (C++/Eigen)

구현 참고: Solà (2018) A micro Lie theory

```cpp
#include <Eigen/Dense>
#include <cmath>

// Skew-symmetric matrix [v]_x
Eigen::Matrix3d skew(const Eigen::Vector3d& v) {
    Eigen::Matrix3d S;
    S <<    0, -v(2),  v(1),
         v(2),     0, -v(0),
        -v(1),  v(0),     0;
    return S;
}

// Right Jacobian Jr(phi), phi in R^3
Eigen::Matrix3d Jr(const Eigen::Vector3d& phi) {
    double n = phi.norm();  // ||phi||
    if (n < 1e-7) {
        // Small-angle: Jr ~ I - 0.5 * [phi]_x
        return Eigen::Matrix3d::Identity() - 0.5 * skew(phi);
    }
    Eigen::Matrix3d Sx = skew(phi);   // [phi]_x (NOT phi/n)
    // Jr = I - (1-cos(n))/n^2 * [phi]_x + (n-sin(n))/n^3 * [phi]_x^2
    return Eigen::Matrix3d::Identity()
         - (1.0 - std::cos(n)) / (n * n) * Sx
         + (n - std::sin(n)) / (n * n * n) * Sx * Sx;
}

// Inverse: Jr_inv(phi)
Eigen::Matrix3d Jr_inv(const Eigen::Vector3d& phi) {
    double n = phi.norm();
    if (n < 1e-7) {
        return Eigen::Matrix3d::Identity() + 0.5 * skew(phi);
    }
    Eigen::Matrix3d Sx = skew(phi);
    double cot_half = std::cos(n/2.0) / std::sin(n/2.0);
    Eigen::Vector3d ph = phi / n;
    return (n/2.0)*cot_half * Eigen::Matrix3d::Identity()
         + (1.0 - (n/2.0)*cot_half) * (ph * ph.transpose())
         + (n/2.0) * skew(ph);
}
```

주의
$\phi < 10^{-7}$: 소각 분기 필수 (0 나누기 방지). 임계값은 float(10^{-4}) vs double(10^{-7}) 구분.

실습 코드: Right Jacobian 구현 (Python/NumPy)


```python
import numpy as np

def skew(v):
    """3-vector -> 3x3 skew-symmetric matrix"""
    return np.array([[ 0,    -v[2],  v[1]],
                     [ v[2],  0,    -v[0]],
                     [-v[1],  v[0],  0   ]])

def Jr(phi):
    """Right Jacobian of SO(3). phi: (3,) array"""
    n = np.linalg.norm(phi)
    if n < 1e-7:
        return np.eye(3) - 0.5 * skew(phi)
    Sx = skew(phi)
    return (np.eye(3)
            - (1 - np.cos(n)) / n**2 * Sx
            + (n - np.sin(n)) / n**3 * (Sx @ Sx))

def Jr_inv(phi):
    """Inverse Right Jacobian of SO(3). phi: (3,) array"""
    n = np.linalg.norm(phi)
    if n < 1e-7:
        return np.eye(3) + 0.5 * skew(phi)
    ph = phi / n
    cot_half = np.cos(n/2) / np.sin(n/2)
    return ((n/2)*cot_half * np.eye(3)
            + (1 - (n/2)*cot_half) * np.outer(ph, ph)
            + (n/2) * skew(ph))

# Quick test
phi = np.array([0.1, 0.2, 0.3])
J  = Jr(phi)
Ji = Jr_inv(phi)
print("Jr @ Jr_inv approx I:", np.allclose(J @ Ji, np.eye(3)))
```

검증
Jr @ Jr_inv $\approx I$: 역행렬 관계 확인. $\phi \to 0$ 극한에서 $J_r \to I$ 확인 가능.

Forster 2017 TRO에서의 역할 (1): Preintegration 공분산

Forster et al. (2017) “On-Manifold Preintegration for Real-Time VI-Odometry”, TRO
핵심 기여: IMU preintegration을 SO(3) 다양체 위에서 수행하여
linearization 없이 정확한 공분산 전파를 구현

Forster 2017 TRO에서의 역할 (1): Preintegration 공분산

Forster et al. (2017) “On-Manifold Preintegration for Real-Time VI-Odometry”, TRO
핵심 기여: IMU preintegration을 SO(3) 다양체 위에서 수행하여
linearization 없이 정확한 공분산 전파를 구현

Preintegration 이산 공분산 전파 (Eq. (44) in paper)

$$\Sigma_{k+1} = A_k \, \Sigma_k \, A_k^{\top} + B_k \, Q_d \, B_k^{\top}$$

여기서 transition matrix A_k의 (1,1) 블록 (회전 부분):

$$A_k^{(11)} = \mathrm{Exp}(-\hat{\boldsymbol{\omega}}_k \, \Delta t)^{\top}, \qquad A_k^{(12)} = -\Delta R_k \, [\hat{\boldsymbol{a}}_k \, \Delta t]_{\times} \, J_r^{(k)}$$

noise 입력 행렬 B_k의 (1,1) 블록:

$$B_k^{(11)} = \Delta R_k \, J_r(\hat{\boldsymbol{\omega}}_k \, \Delta t) \, \Delta t$$

Forster 2017 TRO에서의 역할 (1): Preintegration 공분산

Forster et al. (2017) “On-Manifold Preintegration for Real-Time VI-Odometry”, TRO
핵심 기여: IMU preintegration을 SO(3) 다양체 위에서 수행하여
linearization 없이 정확한 공분산 전파를 구현

Preintegration 이산 공분산 전파 (Eq. (44) in paper)

$$\Sigma_{k+1} = A_k \, \Sigma_k \, A_k^\top + B_k \, Q_d \, B_k^\top$$

여기서 transition matrix A_k의 (1,1) 블록 (회전 부분):

$$A_k^{(11)} = \mathrm{Exp}(-\hat{\boldsymbol{\omega}}_k \, \Delta t)^\top, \qquad A_k^{(12)} = -\Delta R_k \, [\hat{\boldsymbol{a}}_k \, \Delta t]_\times \, J_r^{(k)}$$

noise 입력 행렬 B_k의 (1,1) 블록:

$$B_k^{(11)} = \Delta R_k \, J_r(\hat{\boldsymbol{\omega}}_k \, \Delta t) \, \Delta t$$

해석

- $J_r(\hat{\omega}\Delta t)$: 한 스텝 회전각에서의 Right Jacobian
- Δt가 작으면 $J_r \approx I$ (소각 근사), $B_k \approx \Delta R_k \cdot \Delta t$
- 하지만 정확한 구현에서는 폐쇄형 J_r 사용 권장

Forster 2017 TRO에서의 역할 (2): 바이어스 Jacobian

바이어스 업데이트 필요성
IMU 바이어스 $\boldsymbol{b}_g$가 변하면 preintegration ΔR_{ij}를 다시 적분해야 함.
대신, **1차 보정**(linearization):

$$\Delta\tilde{R}_{ij}(b_g + \delta b_g) \approx \Delta\tilde{R}_{ij}(b_g)\,\mathrm{Exp}\left(\frac{\partial \Delta\tilde{R}_{ij}}{\partial b_g}\,\delta b_g\right)$$

Forster 2017 TRO에서의 역할 (2): 바이어스 Jacobian

바이어스 업데이트 필요성
IMU 바이어스 $\boldsymbol{b}_g$가 변하면 preintegration ΔR_{ij}를 다시 적분해야 함.
대신, **1차 보정**(linearization):

$$\Delta\tilde{R}_{ij}(b_g + \delta b_g) \approx \Delta\tilde{R}_{ij}(b_g)\,\mathrm{Exp}\left(\frac{\partial \Delta\tilde{R}_{ij}}{\partial b_g}\,\delta b_g\right)$$

바이어스 Jacobian (Forster 2017, Eq. (46))

$$\frac{\partial \Delta\tilde{R}_{ij}}{\partial b_g} = -\sum_{k=i}^{j-1} \Delta\tilde{R}_{k+1,j}^{\top} \cdot J_r(\hat{\boldsymbol{\omega}}_k\,\Delta t)\;\Delta t$$

각 스텝의 Right Jacobian의 가중합으로 표현됨

Forster 2017 TRO에서의 역할 (2): 바이어스 Jacobian

바이어스 업데이트 필요성
IMU 바이어스 $\boldsymbol{b}_g$가 변하면 preintegration ΔR_{ij}를 다시 적분해야 함.
대신, **1차 보정**(linearization):

$$\Delta\tilde{R}_{ij}(b_g + \delta b_g) \approx \Delta\tilde{R}_{ij}(b_g)\,\mathrm{Exp}\left(\frac{\partial \Delta\tilde{R}_{ij}}{\partial b_g}\,\delta b_g\right)$$

바이어스 Jacobian (Forster 2017, Eq. (46))

$$\frac{\partial \Delta\tilde{R}_{ij}}{\partial b_g} = -\sum_{k=i}^{j-1} \Delta\tilde{R}_{k+1,j}^{\top} \cdot J_r(\hat{\boldsymbol{\omega}}_k\,\Delta t)\;\Delta t$$

각 스텝의 Right Jacobian의 가중합으로 표현됨

물리적 의미
b_g가 δb_g만큼 변하면:
매 스텝에서 회전 방향이 $J_r\,\Delta t\,\delta b_g$만큼 틀어지고
이후 구간의 회전 ΔR로 propagate됨

실용적 구현
preintegration 시 $J_r^{(k)}$를 함께 누적:

$$\frac{\partial \Delta\tilde{R}}{\partial b_g} \mathrel{+}= -\Delta R_{k+1}^{\top} J_r^{(k)} \Delta t$$

소각 근사: $J_r^{(k)} \approx I$

Forster 2017 TRO에서의 역할 (3): Factor Graph Residual

Preintegration Factor Residual (회전 성분)

$$\boldsymbol{r}_{\Delta R} = \mathrm{Log}\Big(\Delta\tilde{R}_{ij}^{\top} \cdot \boldsymbol{R}_i^{\top}\boldsymbol{R}_j\Big) \in \mathbb{R}^3$$

이 residual은 **SO(3) Log map**으로 so(3)에 투영된 회전 오차

Forster 2017 TRO에서의 역할 (3): Factor Graph Residual

Preintegration Factor Residual (회전 성분)

$$\boldsymbol{r}_{\Delta R} = \mathrm{Log}\Big(\Delta\tilde{R}_{ij}^{\top} \cdot \boldsymbol{R}_i^{\top}\boldsymbol{R}_j\Big) \in \mathbb{R}^3$$

이 residual은 **SO(3) Log map**으로 so(3)에 투영된 회전 오차

Jacobian of $\boldsymbol{r}_{\Delta R}$ w.r.t. $\boldsymbol{R}_j$

$$\frac{\partial \boldsymbol{r}_{\Delta R}}{\partial \delta\boldsymbol{\phi}_j} = J_l^{-1}(\boldsymbol{r}_{\Delta R}) \approx I \quad \text{(residual 소각 시)}$$

J_l^{-1}: Log map의 Jacobian (BCH 역관계로 유도)

Forster 2017 TRO에서의 역할 (3): Factor Graph Residual

Preintegration Factor Residual (회전 성분)

$$\boldsymbol{r}_{\Delta R} = \mathrm{Log}\Big(\Delta\tilde{R}_{ij}^{\top} \cdot \boldsymbol{R}_i^{\top}\boldsymbol{R}_j\Big) \in \mathbb{R}^3$$

이 residual은 **SO(3) Log map**으로 so(3)에 투영된 회전 오차

Jacobian of $\boldsymbol{r}_{\Delta R}$ w.r.t. $\boldsymbol{R}_j$

$$\frac{\partial \boldsymbol{r}_{\Delta R}}{\partial \delta\boldsymbol{\phi}_j} = J_l^{-1}(\boldsymbol{r}_{\Delta R}) \approx I \quad \text{(residual 소각 시)}$$

J_l^{-1}: Log map의 Jacobian (BCH 역관계로 유도)

GTSAM / g2o 구현에서의 처리

- 실제 구현에서는 $\boldsymbol{r}_{\Delta R}$이 수렴 근방에서 소각
- $\Rightarrow J_l^{-1} \approx I$ 근사로 단순화
- 하지만 정밀한 구현(GTSAM NavState)에서는 정확한 J_l^{-1} 사용
- **Chirikjian (2012)**: $\frac{\partial \mathrm{Log}(\boldsymbol{A}^{\top}\boldsymbol{B})}{\partial \boldsymbol{B}}$의 일반 공식

J_r와 Log/Exp 미분의 연결

Exp 미분
$\boldsymbol{R}(\boldsymbol{\phi}) = \mathrm{Exp}(\boldsymbol{\phi})$에 대해:

$$\frac{\partial \mathrm{Exp}(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} = \boldsymbol{R}\,[\,\cdot\,]_\times\; J_r(\boldsymbol{\phi})$$

정확히는: $d\boldsymbol{R} = \boldsymbol{R}\,[J_r\, d\boldsymbol{\phi}]_\times$ (접선공간에서의 differential)

J_r와 Log/Exp 미분의 연결

Exp 미분
$\boldsymbol{R}(\boldsymbol{\phi}) = \text{Exp}(\boldsymbol{\phi})$에 대해:

$$\frac{\partial \text{Exp}(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} = \boldsymbol{R}\,[\,\cdot\,]_\times\; J_r(\boldsymbol{\phi})$$

정확히는: $d\boldsymbol{R} = \boldsymbol{R}\,[J_r\, d\boldsymbol{\phi}]_\times$ (접선공간에서의 differential)

Log 미분
$\boldsymbol{\phi}(\boldsymbol{R}) = \text{Log}(\boldsymbol{R})$에 대해:

$$\frac{\partial \text{Log}(\boldsymbol{R})}{\partial \boldsymbol{R}} = J_r^{-1}(\text{Log}(\boldsymbol{R}))$$

즉, Log map의 Jacobian이 J_r^{-1}

J_r와 Log/Exp 미분의 연결

Exp 미분
$\boldsymbol{R}(\boldsymbol{\phi}) = \mathrm{Exp}(\boldsymbol{\phi})$에 대해:

$$\frac{\partial \mathrm{Exp}(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} = \boldsymbol{R}\,[\,\cdot\,]_\times\, J_r(\boldsymbol{\phi})$$

정확히는: $d\boldsymbol{R} = \boldsymbol{R}\,[J_r\, d\boldsymbol{\phi}]_\times$ (접선공간에서의 differential)

Log 미분
$\boldsymbol{\phi}(\boldsymbol{R}) = \mathrm{Log}(\boldsymbol{R})$에 대해:

$$\frac{\partial \mathrm{Log}(\boldsymbol{R})}{\partial \boldsymbol{R}} = J_r^{-1}(\mathrm{Log}(\boldsymbol{R}))$$

즉, Log map의 Jacobian이 J_r^{-1}

활용: Rp Jacobian (정확한 버전)
$f(\boldsymbol{R}) = \boldsymbol{R}\boldsymbol{p}$에 대해 right perturbation Jacobian:

$$\left.\frac{\partial(\boldsymbol{R}\boldsymbol{p})}{\partial \boldsymbol{\phi}}\right|_{\text{right}} = -\boldsymbol{R}\,[\boldsymbol{p}]_\times$$

이것은 J_r과 무관! 이미 접선공간 방향으로 미분했기 때문.

단, $\frac{\partial \mathrm{Log}(\boldsymbol{R}^\top \boldsymbol{R}')}{\partial \boldsymbol{R}'}$처럼 **Log가 포함된 Jacobian**에는 J_r^{-1} 필요

Retraction과 J_r의 기하학적 의미

Retraction (수축사상)
Lie 군에서 접선벡터를 군 원소로 매핑하는 일반 개념:

$$\text{Retr}_{\boldsymbol{R}}(\delta\boldsymbol{v}) = \boldsymbol{R}\,\text{Exp}(\delta\boldsymbol{v}) \quad (\text{SO(3) exponential retraction})$$

Riemannian geometry에서 “측지선(geodesic)을 따라 이동”

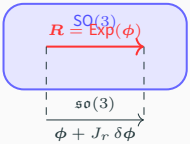
Retraction과 J_r의 기하학적 의미

Retraction (수축사상)
Lie 군에서 접선벡터를 군 원소로 매핑하는 일반 개념:

$$\text{Retr}_{\boldsymbol{R}}(\delta\boldsymbol{v}) = \boldsymbol{R}\,\text{Exp}(\delta\boldsymbol{v}) \quad (\text{SO(3) exponential retraction})$$

Riemannian geometry에서 "측지선(geodesic)을 따라 이동"

J_r의 기하학적 역할



J_r: so(3)의 직선 이동을 SO(3) 곡면 위 이동으로 변환하는 "왜곡 행렬"

직관적 이해

- 평면에서: δx만큼 이동 = δx
- 구면에서: 법선 방향 이동은 없고, 접선 방향 이동만 — 그 "변환계수"가 J_r
- SO(3)에서: 각도가 클수록 J_r이 단위행렬에서 멀어짐 (공간이 더 많이 휨)
- $\phi \to 0$: 평탄해짐 $\Rightarrow J_r \to I$

SO(3) Jacobian과 쿼터니언(Quaternion) 비교

쿼터니언 표현에서의 상황
단위 쿼터니언 $\boldsymbol{q} \in S^3$ ($\|\boldsymbol{q}\| = 1$):

$$\boldsymbol{q}(\phi) = \begin{pmatrix} \cos\frac{\phi}{2} \\ \hat{\boldsymbol{\phi}} \sin\frac{\phi}{2} \end{pmatrix}, \quad \boldsymbol{R}(\boldsymbol{q}) = (q_w^2 - \boldsymbol{q}_v^\top \boldsymbol{q}_v) I + 2\boldsymbol{q}_v \boldsymbol{q}_v^\top + 2q_w \left[\boldsymbol{q}_v\right]_\times$$

SO(3) Jacobian과 쿼터니언(Quaternion) 비교

쿼터니언 표현에서의 상황
단위 쿼터니언 $\boldsymbol{q} \in S^3$ ($\|\boldsymbol{q}\| = 1$):

$$\boldsymbol{q}(\phi) = \begin{pmatrix} \cos\frac{\phi}{2} \\ \hat{\boldsymbol{\phi}} \sin\frac{\phi}{2} \end{pmatrix}, \quad \boldsymbol{R}(\boldsymbol{q}) = (q_w^2 - \boldsymbol{q}_v^\top \boldsymbol{q}_v) I + 2\boldsymbol{q}_v \boldsymbol{q}_v^\top + 2q_w \left[\boldsymbol{q}_v\right]_\times$$

쿼터니언 Jacobian $\partial \boldsymbol{q} / \partial \boldsymbol{\phi}$

$$\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\phi}} = \frac{1}{2} \begin{pmatrix} -\boldsymbol{q}_v^\top \\ q_w I + [\boldsymbol{q}_v]_\times \end{pmatrix} \in \mathbb{R}^{4\times 3}$$

4차원 공간 $\subset \mathbb{R}^4$의 3차원 접선공간으로의 projection

SO(3)와의 관계
$\boldsymbol{R}(\boldsymbol{q})$ Jacobian을 쿼터니언 경유로 계산:

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\phi}} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{q}} \cdot \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\phi}}$$

최종 결과는 SO(3) Jacobian과 동일
(같은 기하학, 다른 표현)

SO(3) Jacobian과 쿼터니언(Quaternion) 비교

쿼터니언 표현에서의 상황
단위 쿼터니언 $\boldsymbol{q} \in S^3$ ($\|\boldsymbol{q}\| = 1$):

$$\boldsymbol{q}(\phi) = \begin{pmatrix} \cos\frac{\phi}{2} \\ \hat{\boldsymbol{\phi}} \sin\frac{\phi}{2} \end{pmatrix}, \quad \boldsymbol{R}(\boldsymbol{q}) = (q_w^2 - \boldsymbol{q}_v^\top \boldsymbol{q}_v) I + 2\boldsymbol{q}_v \boldsymbol{q}_v^\top + 2q_w \left[\boldsymbol{q}_v\right]_\times$$

쿼터니언 Jacobian $\partial \boldsymbol{q} / \partial \boldsymbol{\phi}$

$$\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\phi}} = \frac{1}{2} \begin{pmatrix} -\boldsymbol{q}_v^\top \\ q_w I + [\boldsymbol{q}_v]_\times \end{pmatrix} \in \mathbb{R}^{4\times 3}$$

4차원 공간 $\subset \mathbb{R}^4$의 3차원 접선공간으로의 projection

SO(3)와의 관계
$\boldsymbol{R}(\boldsymbol{q})$ Jacobian을 쿼터니언 경유로 계산:

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\phi}} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{q}} \cdot \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\phi}}$$

최종 결과는 SO(3) Jacobian과 동일
(같은 기하학, 다른 표현)

실용 결론

- 쿼터니언 기반 코드에서도 결국 J_r, J_l이 등장함
- GTSAM의 `Rot3`, Ceres의 `QuaternionManifold` 모두 내부에서 동일한 Jacobian 계산
- 표현만 다를 뿐, **수학적 내용은 동일**

Solà (2018) 레퍼런스 — A micro Lie theory

논문 정보

- **제목**: “A micro Lie theory for state estimation in robotics”
- **저자**: Joan Solà, Jeremie Deray, Dinesh Atchuthan
- **출처**: arXiv:1812.01537 (2018) – 누구나 무료 접근 가능
- **형식**: 58페이지 기술보고서 + 부록 (SLAM 커뮤니티 필독서)

Solà (2018) 레퍼런스 — A micro Lie theory

논문 정보

- **제목**: “A micro Lie theory for state estimation in robotics”
- **저자**: Joan Solà, Jeremie Deray, Dinesh Atchuthan
- **출처**: arXiv:1812.01537 (2018) – 누구나 무료 접근 가능
- **형식**: 58페이지 기술보고서 + 부록 (SLAM 커뮤니티 필독서)

다루는 내용

- SO(2), SO(3), SE(2), SE(3), Sim(3) 위의 Jacobian 통합 정리
- Right/Left Jacobian 폐쇄형
- BCH 근사, Adjoint
- Perturbation calculus 규칙 표 (Table 1)
- SLAM 응용 예시

핵심 Table (Solà 2018)

공식	내용
$\mathrm{Exp}(\phi+\delta) = \mathrm{Exp}(\phi)\mathrm{Exp}(J_r\delta)$	섭동 전파
$J_l = R\,J_r$	Left-Right 관계
$J_r(-\phi) = J_l(\phi)$	부호 반전
$J_r^{-1} \approx I + \frac{1}{2}[\phi]_\times$	소각 역행렬

Solà (2018) 레퍼런스 — A micro Lie theory

논문 정보

- **제목**: “A micro Lie theory for state estimation in robotics”
- **저자**: Joan Solà, Jeremie Deray, Dinesh Atchuthan
- **출처**: arXiv:1812.01537 (2018) – 누구나 무료 접근 가능
- **형식**: 58페이지 기술보고서 + 부록 (SLAM 커뮤니티 필독서)

다루는 내용

- SO(2), SO(3), SE(2), SE(3), Sim(3) 위의 Jacobian 통합 정리
- Right/Left Jacobian 폐쇄형
- BCH 근사, Adjoint
- Perturbation calculus 규칙 표 (Table 1)
- SLAM 응용 예시

핵심 Table (Solà 2018)

공식	내용
$\mathrm{Exp}(\phi+\delta)=\mathrm{Exp}(\phi)\mathrm{Exp}(J_r\delta)$	섭동 전파
$J_l = R\,J_r$	Left-Right 관계
$J_r(-\phi)=J_l(\phi)$	부호 반전
$J_r^{-1}\approx I+\frac{1}{2}[\phi]_\times$	소각 역행렬

권장 사항

이 자료 하나로 SO(3)/SE(3) Jacobian 문제는 대부분 해결된다.
robotics SLAM 연구자라면 반드시 읽어야 할 기준 문서.

Solà (2018) 핵심 표: Lie 군 비교

Lie 군들의 SO(3) Jacobian 구조 비교 (Solà 2018 기반)

군	차원	J_r 크기	소각 근사 J_r
$SO(2)$	1	1×1	1 (스칼라)
$SO(3)$	3	3×3	$I - \frac{1}{2}[\phi]_\times$
$SE(2)$	3	3×3	$I - \frac{1}{2}[\xi]_\times$
$SE(3)$	6	6×6	$I - \frac{1}{2}[\xi]_\times^{(6)}$
$\mathrm{Sim}(3)$	7	7×7	복잡한 구조

Solà (2018) 핵심 표: Lie 군 비교

Lie 군들의 SO(3) Jacobian 구조 비교 (Solà 2018 기반)

군	차원	J_r 크기	소각 근사 J_r
$SO(2)$	1	1×1	1 (스칼라)
$SO(3)$	3	3×3	$I - \frac{1}{2}[\phi]_\times$
$SE(2)$	3	3×3	$I - \frac{1}{2}[\xi]_\times$
$SE(3)$	6	6×6	$I - \frac{1}{2}[\xi]_\times^{(6)}$
$\mathrm{Sim}(3)$	7	7×7	복잡한 구조

SO(3) 주요 항등식 정리

항등식	설명
$\mathrm{Exp}(\phi)\mathrm{Exp}(\delta) = \mathrm{Exp}(\phi + J_l^{-1}\delta)$ (소각 δ)	BCH 1차
$\mathrm{Exp}(\delta)\mathrm{Exp}(\phi) = \mathrm{Exp}(\phi + J_r^{-1}\delta)$ (소각 δ)	BCH 1차
$R\,\mathrm{Exp}(\phi)\,R^\top = \mathrm{Exp}(R\phi)$	Adjoint
$J_l(\phi) = R\,J_r(\phi),\ R = \mathrm{Exp}(\phi)$	L-R 관계
$J_r(\phi)^\top = J_r(-\phi) = J_l(\phi)^{-\top}$	전치 관계

SE(3) Jacobian 간략 소개

SE(3): 위치+회전 함께

$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{pmatrix} \in \mathrm{SE}(3)$, 6D tangent: $\boldsymbol{\xi} = \begin{pmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{pmatrix} \in \mathbb{R}^6$ (translation + rotation)

$$\mathrm{Exp}(\boldsymbol{\xi}) = \begin{pmatrix} \mathrm{Exp}(\boldsymbol{\phi}) & J_l(\boldsymbol{\phi})\boldsymbol{\rho} \\ \boldsymbol{0} & 1 \end{pmatrix}$$

SE(3) Jacobian 간략 소개

SE(3): 위치+회전 함께

$T = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \in \mathrm{SE}(3)$, 6D tangent: $\xi = \begin{pmatrix} \rho \\ \phi \end{pmatrix} \in \mathbb{R}^6$ (translation + rotation)

$$\mathrm{Exp}(\xi) = \begin{pmatrix} \mathrm{Exp}(\phi) & J_l(\phi)\rho \\ 0 & 1 \end{pmatrix}$$

SE(3) Adjoint

$$\mathrm{Ad}_T = \begin{pmatrix} R & [t]_\times R \\ 0 & R \end{pmatrix} \in \mathbb{R}^{6\times 6}$$

6D twist(속도 스크류)를 한 좌표계에서 다른 좌표계로 변환할 때 사용

SE(3) Jacobian 간략 소개

SE(3): 위치+회전 함께

$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{pmatrix} \in \mathrm{SE}(3)$, 6D tangent: $\boldsymbol{\xi} = \begin{pmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{pmatrix} \in \mathbb{R}^6$ (translation + rotation)

$$\mathrm{Exp}(\boldsymbol{\xi}) = \begin{pmatrix} \mathrm{Exp}(\boldsymbol{\phi}) & J_l(\boldsymbol{\phi})\boldsymbol{\rho} \\ \boldsymbol{0} & 1 \end{pmatrix}$$

SE(3) Adjoint

$$\mathrm{Ad}_{\boldsymbol{T}} = \begin{pmatrix} \boldsymbol{R} & [\boldsymbol{t}]_\times \boldsymbol{R} \\ \boldsymbol{0} & \boldsymbol{R} \end{pmatrix} \in \mathbb{R}^{6\times 6}$$

6D twist(속도 스크류)를 한 좌표계에서 다른 좌표계로 변환할 때 사용

SE(3) Right Jacobian

6×6 행렬 (블록 구조):

$$J_r^{SE(3)}(\boldsymbol{\xi}) = \begin{pmatrix} J_r(\boldsymbol{\phi}) & Q(\boldsymbol{\xi}) \\ \boldsymbol{0} & J_r(\boldsymbol{\phi}) \end{pmatrix}$$

9회차 LiDAR 섹션 preview

FAST-LIO, LOAM 등 LiDAR odometry에서:

포인트 $\boldsymbol{p}$의 변환 $\boldsymbol{Tp}$ Jacobian:

$$\left.\frac{\partial(\boldsymbol{Tp})}{\partial\boldsymbol{\xi}}\right| = \begin{pmatrix} I & -[\boldsymbol{Rp}+\boldsymbol{t}]_\times \end{pmatrix}$$

미리보기: 4회차 ESKF에서 J_r이 어떻게 쓰이는가

이 다음 3-4 슬라이드는 4회차의 미리보기

- 4회차 ESKF는 “명목 상태 + 오차 상태”를 분리해서 처리하는 구조
- 그 **오차 상태 전파식**에 J_r이 등장한다 — 이걸 보여주기 위함
- 지금 단계에서 ESKF를 이해할 필요는 없다
  J_r이 “실제로 어디서 쓰이는지” 한 번 보고 넘어가는 게 목적
- 4회차에서 ESKF의 모든 줄을 처음부터 다시 유도한다

미리보기: 4회차 ESKF에서 J_r이 어떻게 쓰이는가

이 다음 3-4 슬라이드는 4회차의 미리보기

- 4회차 ESKF는 “명목 상태 + 오차 상태”를 분리해서 처리하는 구조
- 그 **오차 상태 전파식**에 J_r이 등장한다 — 이걸 보여주기 위함
- 지금 단계에서 ESKF를 이해할 필요는 없다
  J_r이 “실제로 어디서 쓰이는지” 한 번 보고 넘어가는 게 목적
- 4회차에서 ESKF의 모든 줄을 처음부터 다시 유도한다

기억할 한 줄

$$\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi})$$ 의 시간 미분에서 J_r 이 자동으로 등장한다.

학생: 다음 3-4 슬라이드는 빠르게 훑어보고, 4회차에서 다시 만났을 때 “아, 그때 본 것”이라고 알아채면 충분.

(미리보기) ESKF error state dynamics와 J_r

ESKF 오차 상태 전파 (회전 성분)
Nominal state: $\hat{\boldsymbol{R}}_k$, Error state: $\delta\boldsymbol{\phi}_k$ (so(3) 벡터)

자세한 유도는 4회차

$$\boldsymbol{R}_k = \hat{\boldsymbol{R}}_k \,\mathrm{Exp}(\delta\boldsymbol{\phi}_k)$$

한 스텝 propagation:

$$\boldsymbol{R}_{k+1} = \boldsymbol{R}_k \,\mathrm{Exp}(\boldsymbol{\omega}\,\Delta t) \;\approx\; \hat{\boldsymbol{R}}_{k+1} \,\mathrm{Exp}\big(A_\phi\,\delta\boldsymbol{\phi}_k + B_\phi\,\boldsymbol{n}_\omega\big)$$

(미리보기) ESKF error state dynamics와 J_r

ESKF 오차 상태 전파 (회전 성분)
Nominal state: $\hat{\boldsymbol{R}}_k$, Error state: $\delta\boldsymbol{\phi}_k$ (so(3) 벡터)

자세한 유도는 4회차

$$\boldsymbol{R}_k = \hat{\boldsymbol{R}}_k \,\mathrm{Exp}(\delta\boldsymbol{\phi}_k)$$

한 스텝 propagation:

$$\boldsymbol{R}_{k+1} = \boldsymbol{R}_k \,\mathrm{Exp}(\boldsymbol{\omega}\,\Delta t) \;\approx\; \hat{\boldsymbol{R}}_{k+1} \,\mathrm{Exp}\big(A_\phi\,\delta\boldsymbol{\phi}_k + B_\phi\,\boldsymbol{n}_\omega\big)$$

오차 전파 행렬 (J_r 포함)

$$A_\phi = \mathrm{Exp}(-\hat{\boldsymbol{\omega}}\,\Delta t) \approx I - [\hat{\boldsymbol{\omega}}]_\times\,\Delta t$$

$$B_\phi = J_r(\hat{\boldsymbol{\omega}}\,\Delta t)\,\Delta t \approx I\,\Delta t \quad \text{(소각 근사)}$$

오차 공분산:

$$P_{\phi,k+1} = A_\phi\,P_{\phi,k}\,A_\phi^\top + B_\phi\,Q_d\,B_\phi^\top$$

(미리보기) ESKF error state dynamics와 J_r

ESKF 오차 상태 전파 (회전 성분)
Nominal state: $\hat{\boldsymbol{R}}_k$, Error state: $\delta\boldsymbol{\phi}_k$ (so(3) 벡터)

자세한 유도는 4회차

$$\boldsymbol{R}_k = \hat{\boldsymbol{R}}_k \,\mathrm{Exp}(\delta\boldsymbol{\phi}_k)$$

한 스텝 propagation:

$$\boldsymbol{R}_{k+1} = \boldsymbol{R}_k \,\mathrm{Exp}(\boldsymbol{\omega}\,\Delta t) \;\approx\; \hat{\boldsymbol{R}}_{k+1}\,\mathrm{Exp}\big(A_\phi\,\delta\boldsymbol{\phi}_k + B_\phi\,\boldsymbol{n}_\omega\big)$$

오차 전파 행렬 (J_r 포함)

$$A_\phi = \mathrm{Exp}(-\hat{\boldsymbol{\omega}}\,\Delta t) \approx I - [\hat{\boldsymbol{\omega}}]_\times\,\Delta t$$

$$B_\phi = J_r(\hat{\boldsymbol{\omega}}\,\Delta t)\,\Delta t \approx I\,\Delta t \quad \text{(소각 근사)}$$

오차 공분산:

$$P_{\phi,k+1} = A_\phi\,P_{\phi,k}\,A_\phi^\top + B_\phi\,Q_d\,B_\phi^\top$$

실용 구현 수준

- **기본 ESKF**: $J_r \approx I$ (소각 근사), 충분히 정확
- **정밀 구현**: Forster 2017처럼 $J_r(\hat{\boldsymbol{\omega}}\Delta t)$ 폐쇄형 계산
- IMU 100 Hz, $\Delta t = 0.01$ s, $\omega = 10$ rad/s: $\phi = 0.1$ rad ($\approx 5.7°$)
  $\Rightarrow$ 소각 근사 충분히 유효

(미리보기) ESKF 업데이트: inject & reset 단계의 Jacobian

KF 업데이트 후 상태 주입 (State Injection)
추정된 error state $\delta\hat{\boldsymbol{\phi}}$를 nominal state에 합성:

$$\hat{\boldsymbol{R}}^{+} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\hat{\boldsymbol{\phi}})$$

이후 $\delta\hat{\boldsymbol{\phi}} \leftarrow \mathbf{0}$ (reset)

(미리보기) ESKF 업데이트: inject & reset 단계의 Jacobian

KF 업데이트 후 상태 주입 (State Injection)
추정된 error state $\delta\hat{\boldsymbol{\phi}}$를 nominal state에 합성:

$$\hat{\boldsymbol{R}}^{+} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\hat{\boldsymbol{\phi}})$$

이후 $\delta\hat{\boldsymbol{\phi}} \leftarrow \mathbf{0}$ (reset)

Reset 후 공분산 보정: Reset Jacobian G
Reset 전 오차: $\delta\boldsymbol{\phi}_{\text{before}}$, Reset 후 오차: $\delta\boldsymbol{\phi}_{\text{after}}$

$$\delta\boldsymbol{\phi}_{\text{after}} \approx \left(I - \frac{1}{2}\left[\delta\hat{\boldsymbol{\phi}}\right]_{\times}\right)\delta\boldsymbol{\phi}_{\text{before}}$$

Reset Jacobian:

$$G = I - \frac{1}{2}\left[\delta\hat{\boldsymbol{\phi}}\right]_{\times}$$

공분산 업데이트: $P^{+} = G\,(I - \boldsymbol{K}\boldsymbol{H})\,P\,G^{\top}$

(미리보기) ESKF 업데이트: inject & reset 단계의 Jacobian

KF 업데이트 후 상태 주입 (State Injection)
추정된 error state $\delta\hat{\boldsymbol{\phi}}$를 nominal state에 합성:

$$\hat{\boldsymbol{R}}^{+} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\hat{\boldsymbol{\phi}})$$

이후 $\delta\hat{\boldsymbol{\phi}} \leftarrow \mathbf{0}$ (reset)

Reset 후 공분산 보정: Reset Jacobian G
Reset 전 오차: $\delta\boldsymbol{\phi}_{\text{before}}$, Reset 후 오차: $\delta\boldsymbol{\phi}_{\text{after}}$

$$\delta\boldsymbol{\phi}_{\text{after}} \approx \left(I - \frac{1}{2}\left[\delta\hat{\boldsymbol{\phi}}\right]_{\times}\right)\delta\boldsymbol{\phi}_{\text{before}}$$

Reset Jacobian:

$$G = I - \frac{1}{2}\left[\delta\hat{\boldsymbol{\phi}}\right]_{\times}$$

공분산 업데이트: $P^{+} = G\,(I - \boldsymbol{K}\boldsymbol{H})\,P\,G^{\top}$

출처 및 유도

- Roumeliotis & Burdick (2002), Mourikis & Roumeliotis (2007)
- Joan Solà 박사 논문 (2010) “Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot”
- G는 사실상 $J_r^{-1}(\delta\hat{\boldsymbol{\phi}})$의 1차 근사

Right Jacobian의 고유값 분석

$J_r(\boldsymbol{\phi})$의 행렬 구조
$\boldsymbol{\phi} = \phi\,\hat{\boldsymbol{n}}$ ($\hat{\boldsymbol{n}}$: 단위벡터)로 쓰면:

$$J_r(\phi\hat{\boldsymbol{n}}) = \frac{\sin\phi}{\phi} P_\perp + \frac{1-\cos\phi}{\phi}\hat{\boldsymbol{n}}_\times + \hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^\top$$

여기서 $P_\perp = I - \hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^\top$: $\hat{\boldsymbol{n}}$에 수직인 투영

Right Jacobian의 고유값 분석

$J_r(\boldsymbol{\phi})$의 행렬 구조
$\boldsymbol{\phi} = \phi\hat{\boldsymbol{n}}$ ($\hat{\boldsymbol{n}}$: 단위벡터)로 쓰면:

$$J_r(\phi\hat{\boldsymbol{n}}) = \frac{\sin\phi}{\phi}P_\perp + \frac{1-\cos\phi}{\phi}\hat{\boldsymbol{n}}_\times + \hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^\top$$

여기서 $P_\perp = I - \hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^\top$: $\hat{\boldsymbol{n}}$에 수직인 투영

고유값

- $\hat{\boldsymbol{n}}$ 방향: 고유값 $= 1$ (항상)
- $\hat{\boldsymbol{n}}$에 수직 방향: 복소 고유값 $\frac{\sin\phi}{\phi} \pm i\frac{1-\cos\phi}{\phi}$
- 크기: $\left|\frac{\sin\phi + i(1-\cos\phi)}{\phi}\right| = \frac{2\sin(\phi/2)}{\phi}$

Right Jacobian의 고유값 분석

$J_r(\boldsymbol{\phi})$의 행렬 구조
$\boldsymbol{\phi} = \phi\hat{\boldsymbol{n}}$ ($\hat{\boldsymbol{n}}$: 단위벡터)로 쓰면:

$$J_r(\phi\hat{\boldsymbol{n}}) = \frac{\sin\phi}{\phi}P_\perp + \frac{1-\cos\phi}{\phi}\hat{\boldsymbol{n}}_\times + \hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^\top$$

여기서 $P_\perp = I - \hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^\top$: $\hat{\boldsymbol{n}}$에 수직인 투영

고유값

- $\hat{\boldsymbol{n}}$ 방향: 고유값 $= 1$ (항상)
- $\hat{\boldsymbol{n}}$에 수직 방향: 복소 고유값 $\frac{\sin\phi}{\phi} \pm i\frac{1-\cos\phi}{\phi}$
- 크기: $\left|\frac{\sin\phi + i(1-\cos\phi)}{\phi}\right| = \frac{2\sin(\phi/2)}{\phi}$

$\det(J_r(\phi))$

$$\det(J_r) = \frac{2(1-\cos\phi)}{\phi^2}\cdot 1 = \left(\frac{2\sin(\phi/2)}{\phi}\right)^2$$

$\phi \to 0$: det $\to 1$
$\phi = \pi$: det $\to 0$ (singular!)

Singularity 주의
$\phi = \pi$ (180도 회전): J_r 특이!
$\Rightarrow J_r^{-1}$ 존재하지 않음

실용적으로: $\|\boldsymbol{\phi}\| < \pi$에서만 사용
Gimbal lock과 유사한 문제

J_r와 평행이동 Jacobian의 결합: SE(3) 응용

SE(3) Exp의 평행이동 성분
$\boldsymbol{\xi} = (\boldsymbol{\rho}, \boldsymbol{\phi})^\top$에 대해 $\boldsymbol{T} = \text{Exp}(\boldsymbol{\xi})$이면:

$$\boldsymbol{t} = J_l(\boldsymbol{\phi})\,\boldsymbol{\rho}$$

평행이동 $\boldsymbol{t}$은 $\boldsymbol{\rho}$에 Left Jacobian을 곱한 것

J_r와 평행이동 Jacobian의 결합: SE(3) 응용

SE(3) Exp의 평행이동 성분
$\boldsymbol{\xi} = (\boldsymbol{\rho}, \boldsymbol{\phi})^\top$에 대해 $\boldsymbol{T} = \text{Exp}(\boldsymbol{\xi})$이면:

$$\boldsymbol{t} = J_l(\boldsymbol{\phi})\,\boldsymbol{\rho}$$

평행이동 $\boldsymbol{t}$은 $\boldsymbol{\rho}$에 Left Jacobian을 곱한 것

실용 의미
$\boldsymbol{\phi} \to \mathbf{0}$ (소각 회전): $J_l \to I$
$\Rightarrow \boldsymbol{t} \approx \boldsymbol{\rho}$ (직관적)

큰 회전: J_l이 $\boldsymbol{\rho}$를 휘게 만들어
나선형 운동(screw motion)을 표현

FAST-LIO2에서 사용
LiDAR scan-to-map 오차 Jacobian:

$$\boldsymbol{J} = \begin{pmatrix} \boldsymbol{n}^\top J_l^\top & \boldsymbol{n}^\top [\boldsymbol{T}\boldsymbol{p}]_\times \end{pmatrix}$$

포인트-평면 거리의 6D pose Jacobian
($\boldsymbol{n}$: 평면 법선벡터)

J_r와 평행이동 Jacobian의 결합: SE(3) 응용

SE(3) Exp의 평행이동 성분
$\boldsymbol{\xi} = (\boldsymbol{\rho}, \boldsymbol{\phi})^\top$에 대해 $\boldsymbol{T} = \text{Exp}(\boldsymbol{\xi})$이면:

$$\boldsymbol{t} = J_l(\boldsymbol{\phi})\,\boldsymbol{\rho}$$

평행이동 $\boldsymbol{t}$은 $\boldsymbol{\rho}$에 Left Jacobian을 곱한 것

실용 의미
$\boldsymbol{\phi} \to \mathbf{0}$ (소각 회전): $J_l \to I$
$\Rightarrow \boldsymbol{t} \approx \boldsymbol{\rho}$ (직관적)

큰 회전: J_l이 $\boldsymbol{\rho}$를 휘게 만들어
나선형 운동(screw motion)을 표현

FAST-LIO2에서 사용
LiDAR scan-to-map 오차 Jacobian:

$$\boldsymbol{J} = \begin{pmatrix} \boldsymbol{n}^\top J_l^\top & \boldsymbol{n}^\top [\boldsymbol{T}\boldsymbol{p}]_\times \end{pmatrix}$$

포인트-평면 거리의 6D pose Jacobian
($\boldsymbol{n}$: 평면 법선벡터)

요약

SE(3) Jacobian = SO(3)의 J_r / J_l + 위치-회전 coupling 항 Q
로봇 SLAM의 pose optimization 전체가 이 구조 위에 구축됨

정리: SO(3) Jacobian 공식 모음

Right Jacobian $J_r(\phi)$

$$J_r = I - \frac{1-\cos\phi}{\phi^2}[\boldsymbol{\phi}]_\times + \frac{\phi-\sin\phi}{\phi^3}[\boldsymbol{\phi}]_\times^2 \xrightarrow{\phi\to 0} I - \tfrac{1}{2}[\boldsymbol{\phi}]_\times$$

Right Jacobian Inverse $J_r^{-1}(\phi)$

$$J_r^{-1} = \frac{\phi}{2}\cot\frac{\phi}{2} I + \left(1 - \frac{\phi}{2}\cot\frac{\phi}{2}\right)\hat{\boldsymbol{\phi}}\hat{\boldsymbol{\phi}}^\top + \frac{\phi}{2}[\hat{\boldsymbol{\phi}}]_\times \xrightarrow{\phi\to 0} I + \tfrac{1}{2}[\boldsymbol{\phi}]_\times$$

관계식 모음

$J_l(\boldsymbol{\phi}) = J_r(-\boldsymbol{\phi})$	$J_l = R\,J_r,\ R = \mathrm{Exp}(\boldsymbol{\phi})$
$\mathrm{Exp}(\boldsymbol{\phi}+\delta) = \mathrm{Exp}(\boldsymbol{\phi})\mathrm{Exp}(J_r\delta)$	$\mathrm{Exp}(\boldsymbol{\phi}+\delta) = \mathrm{Exp}(J_l\delta)\mathrm{Exp}(\boldsymbol{\phi})$
$\mathrm{Log}(\mathrm{Exp}(\boldsymbol{a})\mathrm{Exp}(\boldsymbol{b})) \approx \boldsymbol{a} + J_l^{-1}(\boldsymbol{a})\boldsymbol{b}$	$R\mathrm{Exp}(\boldsymbol{\phi})R^\top = \mathrm{Exp}(R\boldsymbol{\phi})$
$\Sigma_{\mathrm{SO(3)}} = J_r Q J_r^\top$	$G_{\mathrm{reset}} = I - \frac{1}{2}[\delta\hat{\boldsymbol{\phi}}]_\times$

연습 문제 (1): Right Jacobian 계산

문제 1
$\boldsymbol{\phi} = (0, 0, \pi/6)^\top$ (z축 30도 회전)일 때:

1. $J_r(\boldsymbol{\phi})$를 폐쇄형 공식으로 계산하시오.
2. 소각 근사 $J_r \approx I - \frac{1}{2}[\boldsymbol{\phi}]_\times$와 비교하시오.
3. Frobenius norm 오차 $\|J_r^{\text{exact}} - J_r^{\text{approx}}\|_F$를 구하시오.

연습 문제 (1): Right Jacobian 계산

문제 1
$\boldsymbol{\phi} = (0, 0, \pi/6)^\top$ (z축 30도 회전)일 때:

1. $J_r(\boldsymbol{\phi})$를 폐쇄형 공식으로 계산하시오.
2. 소각 근사 $J_r \approx I - \frac{1}{2}[\boldsymbol{\phi}]_\times$와 비교하시오.
3. Frobenius norm 오차 $\|J_r^{\text{exact}} - J_r^{\text{approx}}\|_F$를 구하시오.

힌트

$\phi = \pi/6$, $[\boldsymbol{\phi}]_\times = \phi \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$:

$$J_r = I - \frac{1 - \cos(\pi/6)}{(\pi/6)^2}[\boldsymbol{\phi}]_\times + \frac{\pi/6 - \sin(\pi/6)}{(\pi/6)^3}[\boldsymbol{\phi}]_\times^2$$
$$\approx \begin{pmatrix} 0.9589 & -0.2588 & 0 \\ 0.2588 & 0.9589 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(수치 계산 검증: Python/MATLAB 권장)

연습 문제 (2): 섭동 전파

문제 2
$\boldsymbol{R}_0 = \mathrm{Exp}(\boldsymbol{\phi}_0)$, $\boldsymbol{\phi}_0 = (\pi/4, 0, 0)^\top$이고
$\delta\boldsymbol{\phi} = (0.01, 0.01, 0.01)^\top$이라 할 때:

1. $\mathrm{Exp}(\boldsymbol{\phi}_0 + \delta\boldsymbol{\phi})$를 직접 계산하시오.
2. $\boldsymbol{R}_0\,\mathrm{Exp}(J_r(\boldsymbol{\phi}_0)\,\delta\boldsymbol{\phi})$를 계산하시오.
3. 두 결과의 차이를 $\|\cdot\|_F$로 측정하고, 이것이 $O(\|\delta\boldsymbol{\phi}\|^2)$인지 확인하시오.

연습 문제 (2): 섭동 전파

문제 2
$\boldsymbol{R}_0 = \mathrm{Exp}(\boldsymbol{\phi}_0)$, $\boldsymbol{\phi}_0 = (\pi/4, 0, 0)^\top$이고
$\delta\boldsymbol{\phi} = (0.01, 0.01, 0.01)^\top$이라 할 때:

1. $\mathrm{Exp}(\boldsymbol{\phi}_0 + \delta\boldsymbol{\phi})$를 직접 계산하시오.
2. $\boldsymbol{R}_0\,\mathrm{Exp}(J_r(\boldsymbol{\phi}_0)\,\delta\boldsymbol{\phi})$를 계산하시오.
3. 두 결과의 차이를 $\|\cdot\|_F$로 측정하고, 이것이 $O(\|\delta\boldsymbol{\phi}\|^2)$인지 확인하시오.

문제 3: BCH 검증
$\boldsymbol{a} = (\pi/3, 0, 0)^\top$, $\boldsymbol{b} = (0, 0.05, 0)^\top$일 때:

1. $\mathrm{Log}(\mathrm{Exp}(\boldsymbol{a})\mathrm{Exp}(\boldsymbol{b}))$를 수치적으로 계산하시오.
2. BCH 1차 근사 $\boldsymbol{a} + J_l^{-1}(\boldsymbol{a})\boldsymbol{b}$와 비교하시오.
3. 0차 근사 $\boldsymbol{a} + \boldsymbol{b}$와도 비교하시오.

BCH 1차 근사가 왜 더 나은지 설명하시오.

연습 문제 (3): 미리보기 ESKF 적용 (선택)

문제 4: 공분산 변환
초기 회전 오차 공분산 $P_\phi = \sigma^2 I, \sigma = 0.1$ rad이고
$\hat{\phi} = (0.3, 0.2, 0.1)^\top$ rad ($\|\hat{\phi}\| \approx 0.374$ rad)에서:

1. $J_r(\hat{\phi})$를 계산하시오.
2. SO(3) 공분산 $\Sigma_{\mathrm{SO}(3)} = J_r P_\phi J_r^\top$를 계산하시오.
3. $J_r \approx I$ 근사와의 차이를 비교하시오.

연습 문제 (3): 미리보기 ESKF 적용 (선택)

문제 4: 공분산 변환
초기 회전 오차 공분산 $P_\phi = \sigma^2 I, \sigma = 0.1$ rad이고
$\hat{\phi} = (0.3, 0.2, 0.1)^\top$ rad ($\|\hat{\phi}\| \approx 0.374$ rad)에서:

1. $J_r(\hat{\phi})$를 계산하시오.
2. SO(3) 공분산 $\Sigma_{\mathrm{SO}(3)} = J_r P_\phi J_r^\top$를 계산하시오.
3. $J_r \approx I$ 근사와의 차이를 비교하시오.

문제 5: ESKF Reset Jacobian
업데이트 후 $\delta\hat{\phi} = (0.05, -0.03, 0.02)^\top$이 추정되었을 때:

1. Reset Jacobian $G = I - \frac{1}{2}[\delta\hat{\phi}]_\times$를 계산하시오.
2. 이것이 $J_r^{-1}(\delta\hat{\phi})$의 1차 근사임을 보이시오.
3. 업데이트 후 공분산 $P^+ = G(I - KH)PG^\top$에서
   G를 생략했을 때의 오차를 quantify 하시오.

심화: Jacobian of composition $R_1 R_2$

회전 합성의 Jacobian (right perturbation)
$f(\boldsymbol{R}_1, \boldsymbol{R}_2) = \boldsymbol{R}_1 \boldsymbol{R}_2$에서 각각에 대한 Jacobian:

R_1 w.r.t. right perturbation

$$\left.\frac{\partial(\boldsymbol{R}_1 \boldsymbol{R}_2)}{\partial \delta \boldsymbol{\phi}_1}\right|_{\text{right}} = I_{3\times 3}$$

(right perturbation으로 $\boldsymbol{R}_1$ 섭동 시 합성 회전도 같은 방향)

R_2 w.r.t. right perturbation

$$\left.\frac{\partial(\boldsymbol{R}_1 \boldsymbol{R}_2)}{\partial \delta \boldsymbol{\phi}_2}\right|_{\text{right}} = \boldsymbol{R}_1^{\top} \cdot I \cdot \boldsymbol{R}_1 = I$$

(Adjoint에 의해 좌표계 변환)

정확히는: $\text{Ad}_{\boldsymbol{R}_1^{\top}} = \boldsymbol{R}_1^{\top}$

심화: Jacobian of composition $R_1 R_2$

회전 합성의 Jacobian (right perturbation)
$f(\boldsymbol{R}_1, \boldsymbol{R}_2) = \boldsymbol{R}_1 \boldsymbol{R}_2$에서 각각에 대한 Jacobian:

R_1 w.r.t. right perturbation

$$\left.\frac{\partial(\boldsymbol{R}_1 \boldsymbol{R}_2)}{\partial \delta \boldsymbol{\phi}_1}\right|_{\text{right}} = I_{3\times 3}$$

(right perturbation으로 $\boldsymbol{R}_1$ 섭동 시 합성 회전도 같은 방향)

R_2 w.r.t. right perturbation

$$\left.\frac{\partial(\boldsymbol{R}_1 \boldsymbol{R}_2)}{\partial \delta \boldsymbol{\phi}_2}\right|_{\text{right}} = \boldsymbol{R}_1^\top \cdot I \cdot \boldsymbol{R}_1 = I$$

(Adjoint에 의해 좌표계 변환)

정확히는: $\text{Ad}_{\boldsymbol{R}_1^\top} = \boldsymbol{R}_1^\top$

역회전 $\boldsymbol{R}^{-1} = \boldsymbol{R}^\top$의 Jacobian

$$\left.\frac{\partial(\boldsymbol{R}^\top)}{\partial \delta \boldsymbol{\phi}}\right|_{\text{right}} = -I_{3\times 3}$$

(직관: 회전 방향을 뒤집으면 Jacobian 부호도 반전)

심화: Jacobian of composition $R_1 R_2$

회전 합성의 Jacobian (right perturbation)
$f(\boldsymbol{R}_1, \boldsymbol{R}_2) = \boldsymbol{R}_1 \boldsymbol{R}_2$에서 각각에 대한 Jacobian:

R_1 w.r.t. right perturbation

$$\left.\frac{\partial(\boldsymbol{R}_1 \boldsymbol{R}_2)}{\partial \delta\boldsymbol{\phi}_1}\right|_{\text{right}} = I_{3\times 3}$$

(right perturbation으로 $\boldsymbol{R}_1$ 섭동 시 합성 회전도 같은 방향)

R_2 w.r.t. right perturbation

$$\left.\frac{\partial(\boldsymbol{R}_1 \boldsymbol{R}_2)}{\partial \delta\boldsymbol{\phi}_2}\right|_{\text{right}} = \boldsymbol{R}_1^\top \cdot I \cdot \boldsymbol{R}_1 = I$$

(Adjoint에 의해 좌표계 변환)

정확히는: $\text{Ad}_{\boldsymbol{R}_1^\top} = \boldsymbol{R}_1^\top$

역회전 $R^{-1} = R^\top$의 Jacobian

$$\left.\frac{\partial(\boldsymbol{R}^\top)}{\partial \delta\boldsymbol{\phi}}\right|_{\text{right}} = -I_{3\times 3}$$

(직관: 회전 방향을 뒤집으면 Jacobian 부호도 반전)

Factor graph에서의 활용
IMU factor의 rotation residual: $\boldsymbol{r} = \text{Log}(\boldsymbol{R}_i^\top \boldsymbol{R}_j \, \Delta R_{ij}^\top)$

$$\left.\frac{\partial \boldsymbol{r}}{\partial \delta\boldsymbol{\phi}_i}\right|_{\text{right}} = -J_l^{-1}(\boldsymbol{r}), \qquad \left.\frac{\partial \boldsymbol{r}}{\partial \delta\boldsymbol{\phi}_j}\right|_{\text{right}} = J_l^{-1}(\boldsymbol{r})\, \boldsymbol{R}_i^\top \boldsymbol{R}_j \, \Delta R_{ij}^\top \cdot (-1) \cdot \ldots$$

SO(3) Jacobian과 로봇 제어: 연결

운동학적 Jacobian (Geometric Jacobian) 복습
로봇 말단 각속도: $\boldsymbol{\omega} = J_\omega(\boldsymbol{q})\,\dot{\boldsymbol{q}}$
이것과 SO(3) Jacobian은 **다른 개념**이지만 연결됨

SO(3) Jacobian과 로봇 제어: 연결

운동학적 Jacobian (Geometric Jacobian) 복습
로봇 말단 각속도: $\boldsymbol{\omega} = J_\omega(\boldsymbol{q})\,\dot{\boldsymbol{q}}$
이것과 SO(3) Jacobian은 **다른 개념**이지만 연결됨

Lie 대수적 해석
SO(3)에서 body angular velocity $\boldsymbol{\omega}$:

$$\boldsymbol{R}^\top \dot{\boldsymbol{R}} = \left[\boldsymbol{\omega}^b\right]_\times \in \mathfrak{so}(3)$$

$\boldsymbol{\omega}^b = J_r^{-1}(\boldsymbol{\phi})\,\dot{\boldsymbol{\phi}}$ (right body velocity)

$\Rightarrow J_r^{-1}$가 관절 속도 $\dot{\boldsymbol{\phi}}$를
body angular velocity $\boldsymbol{\omega}^b$로 변환

응용: 오일러각 Jacobian
오일러각 (ψ, θ, ϕ)에서도 유사한 구조:

$$\boldsymbol{\omega} = E(\psi, \theta, \phi)\,\dot{\boldsymbol{\Theta}}$$

E: 오일러각 "Jacobian" (비특이 조건: $\theta \neq \pm 90°$)
SO(3) 전역 J_r과 유사한 역할이지만 좌표 의존적

SO(3) Jacobian과 로봇 제어: 연결

운동학적 Jacobian (Geometric Jacobian) 복습
로봇 말단 각속도: $\boldsymbol{\omega} = J_\omega(\boldsymbol{q})\,\dot{\boldsymbol{q}}$
이것과 SO(3) Jacobian은 **다른 개념**이지만 연결됨

Lie 대수적 해석
SO(3)에서 body angular velocity $\boldsymbol{\omega}$:

$$\boldsymbol{R}^\top \dot{\boldsymbol{R}} = \left[\boldsymbol{\omega}^b\right]_\times \in \mathfrak{so}(3)$$

$\boldsymbol{\omega}^b = J_r^{-1}(\boldsymbol{\phi})\,\dot{\boldsymbol{\phi}}$ (right body velocity)

$\Rightarrow J_r^{-1}$가 관절 속도 $\dot{\boldsymbol{\phi}}$를
body angular velocity $\boldsymbol{\omega}^b$로 변환

응용: 오일러각 Jacobian
오일러각 (ψ, θ, ϕ)에서도 유사한 구조:

$$\boldsymbol{\omega} = E(\psi, \theta, \phi)\,\dot{\boldsymbol{\Theta}}$$

E: 오일러각 "Jacobian" (비특이 조건: $\theta \neq \pm 90°$)
SO(3) 전역 J_r과 유사한 역할이지만 좌표 의존적

핵심 차이

SO(3) J_r: 좌표 독립적 (차표계 선택 불필요)
오일러각 E: 특정 좌표계 의존적, singularity 존재
$\Rightarrow$ ESKF에서 J_r 기반 접근이 더 강건

SO(3) Jacobian: 학습 로드맵

이 파트에서 배운 핵심 개념 흐름

1. **동기**: SO(3)는 벡터공간이 아니므로 미분에 접선공간(tangent space) 필요
2. **Right perturbation**: $f(R\mathrm{Exp}(\epsilon))$의 극한 $\rightarrow$ right Jacobian
3. J_r **정의**: 적분 $\rightarrow$ 폐쇄형, $\phi \rightarrow 0$이면 I
4. $J_l = J_r(-\phi)$: Adjoint로 연결
5. J_r^{-1} **폐쇄형**: 소각 근사 $I + \frac{1}{2}[\phi]_\times$
6. **소각 근사**: ESKF 표준, $< 10°$에서 유효
7. **BCH 1차**: $\mathrm{Log}(\mathrm{Exp}a\mathrm{Exp}b) \approx a + J_l^{-1}b$
8. **섭동 전파**: $\mathrm{Exp}(\phi+\delta) = \mathrm{Exp}(\phi)\mathrm{Exp}(J_r\delta)$
9. **공분산 변환**: $\Sigma_{\mathrm{SO}(3)} = J_r Q J_r^\top$
10. **ESKF 적용**: preintegration, inject/reset Jacobian

SO(3) Jacobian: 학습 로드맵

이 파트에서 배운 핵심 개념 흐름

1. **동기**: SO(3)는 벡터공간이 아니므로 미분에 접선공간(tangent space) 필요
2. **Right perturbation**: $f(R\text{Exp}(\epsilon))$의 극한 $\rightarrow$ right Jacobian
3. J_r **정의**: 적분 $\rightarrow$ 폐쇄형, $\phi \rightarrow 0$이면 I
4. $J_l = J_r(-\phi)$: Adjoint로 연결
5. J_r^{-1} **폐쇄형**: 소각 근사 $I + \frac{1}{2}[\phi]_\times$
6. **소각 근사**: ESKF 표준, $< 10^\circ$에서 유효
7. **BCH 1차**: $\text{Log}(\text{Exp}a\text{Exp}b) \approx a + J_l^{-1}b$
8. **섭동 전파**: $\text{Exp}(\phi + \delta) = \text{Exp}(\phi)\text{Exp}(J_r\delta)$
9. **공분산 변환**: $\Sigma_{\text{SO}(3)} = J_r Q J_r^\top$
10. **ESKF 적용**: preintegration, inject/reset Jacobian

다음 강의에서의 사용처

- **4회차**: ESKF 전체 유도에서 J_r 등장
- **5회차**: IMU preintegration 공분산에서 $J_r^{(k)}$
- **7/8회차**: VIO factor graph residual Jacobian
- **9회차**: LiDAR SE(3) Jacobian

참고문헌

- Solà et al. arXiv:1812.01537
- Forster et al. TRO 2017
- Chirikjian (2012) Stochastic Models...
- Barfoot (2017) State Estimation for Robotics

회차 요약: SO(3) Jacobians

5대 핵심 공식

1. $J_r(\phi)$ **정의 (폐쇄형)**:

$$J_r(\phi) = I - \frac{1-\cos\phi}{\phi^2}[\phi]_\times + \frac{\phi-\sin\phi}{\phi^3}[\phi]_\times^2$$

2. **Left-Right 관계**:

$$J_l(\phi) = J_r(-\phi) = \mathrm{Exp}(\phi)\, J_r(\phi)$$

3. **소각 근사**:

$$J_r(\phi) \approx I - \tfrac{1}{2}[\phi]_\times, \quad J_r^{-1}(\phi) \approx I + \tfrac{1}{2}[\phi]_\times$$

4. **섭동 전파** (ESKF/preintegration 핵심):

$$\mathrm{Exp}(\phi + \delta\phi) = \mathrm{Exp}(\phi)\,\mathrm{Exp}(J_r(\phi)\,\delta\phi)$$

5. **공분산 변환**:

$$\Sigma_{\mathrm{SO}(3)} = J_r(\phi)\,\Sigma_{\mathfrak{so}(3)}\,J_r(\phi)^\top$$

다음 회차 예고: 3회차 선형 KF, 4회차 ESKF

2회차 전체 정리

- SO(3) 기초: 군 구조, Exp/Log map, Rodrigues 공식
- Right perturbation: ESKF의 기본 관례
- **SO(3) Jacobians**: J_r, J_l, BCH, 섭동 전파, 공분산 변환

다음 회차 예고: 3회차 선형 KF, 4회차 ESKF

2회차 전체 정리

- SO(3) 기초: 군 구조, Exp/Log map, Rodrigues 공식
- Right perturbation: ESKF의 기본 관례
- **SO(3) Jacobians**: J_r, J_l, BCH, 섭동 전파, 공분산 변환

4회차(ESKF)에서 다시 만날 핵심 도구 — 미리 보기

- Right perturbation: $\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi})$
- 선형화: $\mathrm{Exp}(\delta\boldsymbol{\phi}) \approx I + [\delta\boldsymbol{\phi}]_\times$
- Error dynamics: J_r^{-1} 포함한 F 행렬
- Noise 전파: $J_r\, Q_d\, J_r^\top$
- Update: $\hat{\boldsymbol{R}} \leftarrow \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\hat{\boldsymbol{\phi}})$
- Reset Jacobian: $G = I - \frac{1}{2}[\delta\hat{\boldsymbol{\phi}}]_\times$
- Full ESKF: 15D error state
  $\delta\boldsymbol{x} = (\delta\boldsymbol{\phi}, \delta\boldsymbol{p}, \delta\boldsymbol{v}, \delta\boldsymbol{b}_g, \delta\boldsymbol{b}_a)^\top$

다음 회차 예고: 3회차 선형 KF, 4회차 ESKF

2회차 전체 정리

- SO(3) 기초: 군 구조, Exp/Log map, Rodrigues 공식
- Right perturbation: ESKF의 기본 관례
- **SO(3) Jacobians**: J_r, J_l, BCH, 섭동 전파, 공분산 변환

4회차(ESKF)에서 다시 만날 핵심 도구 — 미리 보기

- Right perturbation: $\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi})$
- 선형화: $\mathrm{Exp}(\delta\boldsymbol{\phi}) \approx I + [\delta\boldsymbol{\phi}]_\times$
- Error dynamics: J_r^{-1} 포함한 F 행렬
- Noise 전파: $J_r\, Q_d\, J_r^\top$
- Update: $\hat{\boldsymbol{R}} \leftarrow \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\hat{\boldsymbol{\phi}})$
- Reset Jacobian: $G = I - \frac{1}{2}[\delta\hat{\boldsymbol{\phi}}]_\times$
- Full ESKF: 15D error state $\delta\boldsymbol{x} = (\delta\boldsymbol{\phi}, \delta\boldsymbol{p}, \delta\boldsymbol{v}, \delta\boldsymbol{b}_g, \delta\boldsymbol{b}_a)^\top$

먼저 3회차 (선형 KF) → 다음 4회차에서 ESKF 전체 유도

참고: Joan Solà (2017) “Quaternion kinematics for the error-state Kalman filter”
arXiv:1711.02508 — ESKF 최고의 입문 자료

3회차: 선형 Kalman Filter

SO(3)는 3D 회전의 매니폴드.
tangent space $\mathfrak{so}(3) \cong \mathbb{R}^3 \to$ error state $\delta\boldsymbol{\phi}$는 제약 없는 벡터.

오늘의 목표:

- 선형 시스템에서 KF의 다섯 줄을 **손으로** 유도
- 각 줄의 왜를 한 마디로 말하기
- 1D 위치+속도 KF Python 30줄로 belief가 춤추는 모습 관찰

선형 시스템의 정의

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{w}_k, \quad \boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0},\, \boldsymbol{Q})$$

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k, \quad \boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0},\, \boldsymbol{R})$$

- $\boldsymbol{F}$: state transition matrix
- $\boldsymbol{B}$: 입력 게인 (선택 — 없으면 0)
- $\boldsymbol{H}$: 측정 행렬
- $\boldsymbol{w}, \boldsymbol{v}$: 가우시안 잡음, 서로 독립

선형 시스템의 정의

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{w}_k, \quad \boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0},\, \boldsymbol{Q})$$

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k, \quad \boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0},\, \boldsymbol{R})$$

- $\boldsymbol{F}$: state transition matrix
- $\boldsymbol{B}$: 입력 게인 (선택 — 없으면 0)
- $\boldsymbol{H}$: 측정 행렬
- $\boldsymbol{w}, \boldsymbol{v}$: 가우시안 잡음, 서로 독립

필수 가정 4개
선형 process, 선형 측정, 가우시안 잡음, 잡음 독립.

왜 이 가정들이 KF를 살리는가

- 선형 변환은 가우시안을 또 가우시안으로 보낸다 (1회차 보조 정리)
- 가우시안의 곱도 가우시안 (1회차 보조 정리)
- 그래서 prior가 가우시안이면 영원히 가우시안
- belief가 **(평균, 공분산)**만으로 완전 기술됨 → 컴퓨터로 풀 수 있음

왜 이 가정들이 KF를 살리는가

- 선형 변환은 가우시안을 또 가우시안으로 보낸다 (1회차 보조 정리)
- 가우시안의 곱도 가우시안 (1회차 보조 정리)
- 그래서 prior가 가우시안이면 영원히 가우시안
- belief가 **(평균, 공분산)**만으로 완전 기술됨 → 컴퓨터로 풀 수 있음

핵심
KF의 5줄은 모두 이 두 보조 정리의 적용에 불과하다.

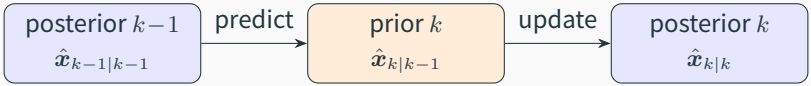
표기 정리 (이번 회차 내내)

- $\hat{\boldsymbol{x}}_{k|k-1}$: $k-1$까지의 측정으로 k를 예측한 평균 (prior)
- $\boldsymbol{P}_{k|k-1}$: 그 분포의 공분산
- $\hat{\boldsymbol{x}}_{k|k}$: k의 측정까지 흡수한 평균 (posterior)
- $\boldsymbol{P}_{k|k}$: 그 공분산

표기 정리 (이번 회차 내내)

- $\hat{\boldsymbol{x}}_{k|k-1}$: $k-1$까지의 측정으로 k를 예측한 평균 (prior)
- $\boldsymbol{P}_{k|k-1}$: 그 분포의 공분산
- $\hat{\boldsymbol{x}}_{k|k}$: k의 측정까지 흡수한 평균 (posterior)
- $\boldsymbol{P}_{k|k}$: 그 공분산

Bayes filter 그림으로 다시:



Predict 유도 (1) — 평균

$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{w}_k$ 의 양변에 기댓값:

$$\mathbb{E}[\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k-1}] \;=\; \boldsymbol{F}\,\mathbb{E}[\boldsymbol{x}_{k-1} \mid \boldsymbol{z}_{1:k-1}] + \boldsymbol{B}\boldsymbol{u}_k$$

Predict 유도 (1) — 평균

$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{w}_k$ 의 양변에 기댓값:

$$\mathbb{E}[\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k-1}] \;=\; \boldsymbol{F}\,\mathbb{E}[\boldsymbol{x}_{k-1} \mid \boldsymbol{z}_{1:k-1}] + \boldsymbol{B}\boldsymbol{u}_k$$

Predict 평균 식

$$\boxed{\hat{\boldsymbol{x}}_{k|k-1} \;=\; \boldsymbol{F}\,\hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}\boldsymbol{u}_k}$$

잡음 $\boldsymbol{w}_k$의 기댓값이 0이라 사라짐. 평균은 단순히 선형으로 옮긴 것.

Predict 유도 (2) — 공분산

편차 정의: $\boldsymbol{e}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}$

$\boldsymbol{e}_k = \boldsymbol{F}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1|k-1}) + \boldsymbol{w}_k = \boldsymbol{F}\boldsymbol{e}_{k-1} + \boldsymbol{w}_k$

편차 정의: $\boldsymbol{e}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}$
$\boldsymbol{e}_k = \boldsymbol{F}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1|k-1}) + \boldsymbol{w}_k = \boldsymbol{F}\boldsymbol{e}_{k-1} + \boldsymbol{w}_k$ 공분산은 $\mathbb{E}[\boldsymbol{e}_k \boldsymbol{e}_k^\top]$:

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\,\mathbb{E}[\boldsymbol{e}_{k-1}\boldsymbol{e}_{k-1}^\top]\,\boldsymbol{F}^\top + \mathbb{E}[\boldsymbol{w}_k \boldsymbol{w}_k^\top]$$

($\boldsymbol{e}_{k-1}$과 $\boldsymbol{w}_k$가 독립이라 교차항 사라짐)

Predict 유도 (2) — 공분산

편차 정의: $\boldsymbol{e}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}$
$\boldsymbol{e}_k = \boldsymbol{F}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1|k-1}) + \boldsymbol{w}_k = \boldsymbol{F}\boldsymbol{e}_{k-1} + \boldsymbol{w}_k$ 공분산은 $\mathbb{E}[\boldsymbol{e}_k \boldsymbol{e}_k^\top]$:

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\,\mathbb{E}[\boldsymbol{e}_{k-1}\boldsymbol{e}_{k-1}^\top]\,\boldsymbol{F}^\top + \mathbb{E}[\boldsymbol{w}_k \boldsymbol{w}_k^\top]$$

($\boldsymbol{e}_{k-1}$과 $\boldsymbol{w}_k$가 독립이라 교차항 사라짐)

Predict 공분산 식

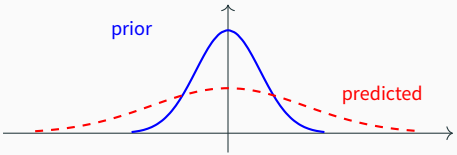
$$\boxed{\boldsymbol{P}_{k|k-1} \;=\; \boldsymbol{F}\,\boldsymbol{P}_{k-1|k-1}\,\boldsymbol{F}^\top + \boldsymbol{Q}}$$

Predict의 의미 한 마디

- **평균**은 dynamics를 따라 흘러간다
- **공분산**은 dynamics에 의해 회전/늘어나고 ($\boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^\top$), process noise만큼 커진다 ($+\boldsymbol{Q}$)
- 직관: ”예측만 하면 belief는 점점 퍼진다”

Predict의 의미 한 마디

- **평균**은 dynamics를 따라 흘러간다
- **공분산**은 dynamics에 의해 회전/늘어나고 ($\boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^\top$), process noise만큼 커진다 ($+\boldsymbol{Q}$)
- 직관: ”예측만 하면 belief는 점점 퍼진다”



Update 유도의 두 갈래

같은 답에 도달하는 두 길:

1. **가우시안 곱셈** 길 (1회차의 1D 결과를 일반화)
2. **최적 추정량** 길 (innovation 잔차의 분산을 최소화)

Update 유도의 두 갈래

같은 답에 도달하는 두 길:

1. **가우시안 곱셈** 길 (1회차의 1D 결과를 일반화)
2. **최적 추정량** 길 (innovation 잔차의 분산을 최소화)

여기선 직관적인 (2)번 길로 간다 — Kalman의 1960년 논문 길이기도 함.

Update 유도 (1) — 추정 형태 가정

posterior 평균을 선형 보정 형태로 가정:

$$\hat{\boldsymbol{x}}_{k|k} \;=\; \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}\left(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1}\right)$$

- $\boldsymbol{K}$: 아직 모르는 게인 행렬
- $\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1}$: **innovation** (혁신, 측정과 예측의 차이)

Update 유도 (1) — 추정 형태 가정

posterior 평균을 선형 보정 형태로 가정:

$$\hat{\boldsymbol{x}}_{k|k} \;=\; \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}\left(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1}\right)$$

- $\boldsymbol{K}$: 아직 모르는 게인 행렬
- $\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1}$: **innovation** (혁신, 측정과 예측의 차이)

질문
posterior의 공분산을 가장 작게 만드는 $\boldsymbol{K}$는?

Update 유도 (2) — posterior 공분산

$\boldsymbol{e}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k}$, $\tilde{\boldsymbol{e}}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}$ (predict 잔차)
innovation: $\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{H}\tilde{\boldsymbol{e}}_k + \boldsymbol{v}_k$

Update 유도 (2) — posterior 공분산

$\boldsymbol{e}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k}$, $\tilde{\boldsymbol{e}}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}$ (predict 잔차)
innovation: $\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{H}\tilde{\boldsymbol{e}}_k + \boldsymbol{v}_k$ 대입:

$$\boldsymbol{e}_k = \tilde{\boldsymbol{e}}_k - \boldsymbol{K}(\boldsymbol{H}\tilde{\boldsymbol{e}}_k + \boldsymbol{v}_k) = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\tilde{\boldsymbol{e}}_k - \boldsymbol{K}\boldsymbol{v}_k$$

Update 유도 (2) — posterior 공분산

$\boldsymbol{e}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k}$, $\tilde{\boldsymbol{e}}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}$ (predict 잔차)
innovation: $\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{H}\tilde{\boldsymbol{e}}_k + \boldsymbol{v}_k$ 대입:

$$\boldsymbol{e}_k = \tilde{\boldsymbol{e}}_k - \boldsymbol{K}(\boldsymbol{H}\tilde{\boldsymbol{e}}_k + \boldsymbol{v}_k) = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\tilde{\boldsymbol{e}}_k - \boldsymbol{K}\boldsymbol{v}_k$$

공분산:

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\,\boldsymbol{P}_{k|k-1}\,(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})^\top + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^\top$$

이게 **Joseph form**. 다음 슬라이드에서 단순화.

Update 유도 (3) — 최적 게인

$\boldsymbol{P}_{k|k}$의 trace를 최소화하는 $\boldsymbol{K}$를 찾자 (총 분산 최소).

$$\frac{\partial}{\partial \boldsymbol{K}} \mathrm{tr}\, \boldsymbol{P}_{k|k} = 0$$

Update 유도 (3) — 최적 게인

$\boldsymbol{P}_{k|k}$의 trace를 최소화하는 $\boldsymbol{K}$를 찾자 (총 분산 최소).

$$\frac{\partial}{\partial \boldsymbol{K}} \operatorname{tr} \boldsymbol{P}_{k|k} = 0$$

편미분 (행렬 미분 공식 활용) 후 정리:

$$-2(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\, \boldsymbol{P}_{k|k-1}\, \boldsymbol{H}^{\top} + 2\boldsymbol{K}\boldsymbol{R} = \boldsymbol{0}$$

Update 유도 (3) — 최적 게인

$\boldsymbol{P}_{k|k}$의 trace를 최소화하는 $\boldsymbol{K}$를 찾자 (총 분산 최소).

$$\frac{\partial}{\partial \boldsymbol{K}} \operatorname{tr} \boldsymbol{P}_{k|k} = 0$$

편미분 (행렬 미분 공식 활용) 후 정리:

$$-2(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\, \boldsymbol{P}_{k|k-1}\, \boldsymbol{H}^{\top} + 2\boldsymbol{K}\boldsymbol{R} = \boldsymbol{0}$$

$\boldsymbol{K}$에 대해 풀면:

$$\boxed{\boldsymbol{K} \;=\; \boldsymbol{P}_{k|k-1}\, \boldsymbol{H}^{\top} \left(\boldsymbol{H}\, \boldsymbol{P}_{k|k-1}\, \boldsymbol{H}^{\top} + \boldsymbol{R}\right)^{-1}}$$

$\boldsymbol{S} = \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^{\top} + \boldsymbol{R}$ — **innovation 공분산**.

Kalman gain의 의미

$$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{H}^{\top}\boldsymbol{S}^{-1}$$

- $\boldsymbol{P}\boldsymbol{H}^{\top}$: state ↔ 측정의 cross-covariance
- $\boldsymbol{S}^{-1}$: 측정 공간에서의 역분산 (= 정밀도)
- 둘의 곱 = ”측정 정보를 state로 끌어오는 변환”

Kalman gain의 의미

$$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{H}^\top \boldsymbol{S}^{-1}$$

- $\boldsymbol{P}\boldsymbol{H}^\top$: state ↔ 측정의 cross-covariance
- $\boldsymbol{S}^{-1}$: 측정 공간에서의 역분산 (= 정밀도)
- 둘의 곱 = "측정 정보를 state로 끌어오는 변환"

한계 사례

- $\boldsymbol{R} \to 0$: 완벽한 측정 → $\boldsymbol{K} \to \boldsymbol{H}^\dagger$ (측정값을 그대로 신뢰)
- $\boldsymbol{R} \to \infty$: 무의미한 측정 → $\boldsymbol{K} \to 0$ (측정 무시, predict만 사용)

Update 유도 (4) — posterior 공분산: Joseph form이 표준

최적 $\boldsymbol{K}$를 대입하면 두 가지 형태:

짧은 형 $\quad \boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\,\boldsymbol{P}_{k|k-1} \quad \leftarrow$ 수학적으로는 같지만 수치 불안정

Update 유도 (4) — posterior 공분산: Joseph form이 표준

최적 $\boldsymbol{K}$를 대입하면 두 가지 형태:

짧은 형 $\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\,\boldsymbol{P}_{k|k-1}$ ← 수학적으로는 같지만 수치 불안정

$$\boxed{\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\,\boldsymbol{P}_{k|k-1}\,(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})^{\top} + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^{\top}} \quad \leftarrow \textbf{Joseph form}$$

Update 유도 (4) — posterior 공분산: Joseph form이 표준

최적 $\boldsymbol{K}$를 대입하면 두 가지 형태:

짧은 형 $\quad \boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\,\boldsymbol{P}_{k|k-1} \quad \leftarrow$ 수학적으로는 같지만 수치 불안정

$$\boxed{\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\,\boldsymbol{P}_{k|k-1}\,(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})^{\top} + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^{\top}} \quad \leftarrow \textbf{Joseph form}$$

Joseph form을 항상 써야 하는 이유
짧은 형은 최적 $\boldsymbol{K}$일 때만 양정치성을 보장.
부동소수점 오차, 부정확한 $\boldsymbol{K}$ 상황에서도 Joseph form은 구조적으로 양정치.
$(\boldsymbol{A}\boldsymbol{P}\boldsymbol{A}^{\top} + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^{\top})$ — 두 항 모두 양반정치의 합 $\Rightarrow$ 항상 양반정치.
실무 규칙: 항상 Joseph form. 짧은 형은 쓰지 않는다.

이걸로 KF 완성 — Joseph form 포함 표준 버전

Predict

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}\hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}\boldsymbol{u}_k$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}^\top + \boldsymbol{Q}$$

Update

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

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

$$\boldsymbol{P}_{k|k} = \underbrace{\left(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}\right)\boldsymbol{P}_{k|k-1}\left(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}\right)^\top + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^\top}_{\text{Joseph form — 항상 이것}}$$

이걸로 KF 완성 — Joseph form 포함 표준 버전

Predict

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}\hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}\boldsymbol{u}_k$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}^\top + \boldsymbol{Q}$$

Update

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

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

$$\boldsymbol{P}_{k|k} = \underbrace{\left(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}\right)\boldsymbol{P}_{k|k-1}\left(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}\right)^\top + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^\top}_{\text{Joseph form — 항상 이것}}$$

외울 필요 없다. 유도할 줄 알면 평생 본인 것. Joseph form만 기억하면 됨.

각 줄을 한 마디로

- $\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}\hat{\boldsymbol{x}}_{k-1|k-1}$: 평균을 dynamics로 옮긴다
- $\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^{\top} + \boldsymbol{Q}$: 공분산을 늘리고, process noise만큼 더 부푼다
- $\boldsymbol{K} = \boldsymbol{P}\boldsymbol{H}^{\top}\boldsymbol{S}^{-1}$: 측정 정밀도와 cross-covariance의 곱 = 측정에서 state로의 게인
- $\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}\,\boldsymbol{r}$: 잔차에 게인을 곱해 평균 보정
- $\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}$: 측정만큼 공분산이 줄어든다

1D KF는 1회차 보조 정리와 같은 식임을 확인

$\boldsymbol{F} = 1, \boldsymbol{H} = 1, \boldsymbol{Q} = q, \boldsymbol{R} = r, \boldsymbol{P}_{k|k-1} = \sigma_p^2$ 대입:

$$K = \frac{\sigma_p^2}{\sigma_p^2 + r}$$

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

$$\sigma_{k|k}^2 = (1 - K)\sigma_p^2 = \frac{\sigma_p^2 \cdot r}{\sigma_p^2 + r}$$

1D KF는 1회차 보조 정리와 같은 식임을 확인

$\boldsymbol{F} = 1, \boldsymbol{H} = 1, \boldsymbol{Q} = q, \boldsymbol{R} = r, \boldsymbol{P}_{k|k-1} = \sigma_p^2$ 대입:

$$K = \frac{\sigma_p^2}{\sigma_p^2 + r}$$

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

$$\sigma_{k|k}^2 = (1 - K)\sigma_p^2 = \frac{\sigma_p^2 \cdot r}{\sigma_p^2 + r}$$

마지막 식을 정리하면 $1/\sigma_{k|k}^2 = 1/\sigma_p^2 + 1/r$ — **정밀도 합**. 1회차의 가우시안 곱과 동일.

공분산이 음이 안 된다는 것

- $\boldsymbol{P}$는 항상 **양 준정치** ($\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} \geq 0$)
- KF가 시작 상태에서 그렇다면 모든 step에서 그래야 함
- Joseph form은 이를 대수적으로 보장
- 짧은 form $(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}$는 산술 오차로 음의 고유값을 만들 수 있음 → 7회차 디테일

공분산이 음이 안 된다는 것

- $\boldsymbol{P}$는 항상 **양 준정치** ($\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} \geq 0$)
- KF가 시작 상태에서 그렇다면 모든 step에서 그래야 함
- Joseph form은 이를 대수적으로 보장
- 짧은 form $(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}$는 산술 오차로 음의 고유값을 만들 수 있음 → 7회차 디테일

practical
실제 코드에서 매 step 후 $\boldsymbol{P} \leftarrow \frac{1}{2}(\boldsymbol{P} + \boldsymbol{P}^\top)$ 같은 대칭화는 거의 공짜 보험.

초기값 $\hat{x}_0, P_0$ 어떻게 잡나

- $\hat{x}_0$: 알고 있는 만큼 — 모르면 0
- P_0: **크게** 잡는다 ("나는 잘 모른다")
- 너무 작게 잡으면 측정이 들어와도 belief가 안 움직임 (과신뢰)
- 너무 크게 잡으면 첫 측정이 거의 완전히 덮어씀 (괜찮음, 보통 안전)

초기값 $\hat{x}_0, P_0$ 어떻게 잡나

- $\hat{x}_0$: 알고 있는 만큼 — 모르면 0
- P_0: **크게** 잡는다 ("나는 잘 모른다")
- 너무 작게 잡으면 측정이 들어와도 belief가 안 움직임 (과신뢰)
- 너무 크게 잡으면 첫 측정이 거의 완전히 덮어씀 (괜찮음, 보통 안전)

Rule of thumb: 알 수 있는 양은 분산 ~ 0, 모르는 양은 분산 $\sim 10^4\times$ 자연 단위

Q, R 튜닝 — 가장 많은 사람이 망하는 곳

- R: 센서 datasheet의 noise spec에서 출발
- Q: 모델이 잡지 못하는 운동의 변동성 — 보수적으로 키워라
- Q가 너무 작으면 → 모델 과신뢰 → 추정이 측정에 안 따라옴
- Q가 너무 크면 → predict가 너무 부풀어 → 측정이 항상 우세 → 노이즈 통과

Q, R 튜닝 — 가장 많은 사람이 망하는 곳

- R: 센서 datasheet의 noise spec에서 출발
- Q: 모델이 잡지 못하는 운동의 변동성 — 보수적으로 키워라
- Q가 너무 작으면 → 모델 과신뢰 → 추정이 측정에 안 따라옴
- Q가 너무 크면 → predict가 너무 부풀어 → 측정이 항상 우세 → 노이즈 통과

Normalized innovation $\nu^\top S^{-1} \nu$
이게 평균적으로 $\dim(z)$ 근처면 KF가 일관성 있다고 본다.
7회차에서 Mahalanobis와 함께 정밀하게.

1D 위치+속도 KF — 모델 세팅

state $\boldsymbol{x} = [x, v]^\top$, 측정 $z = x$ (위치만).

$$\boldsymbol{F} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}, \quad \boldsymbol{H} = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

$$\boldsymbol{Q} = \begin{pmatrix} q_x & 0 \\ 0 & q_v \end{pmatrix}, \quad \boldsymbol{R} = r$$

1D 위치+속도 KF — 모델 세팅

state $\boldsymbol{x} = [x, v]^\top$, 측정 $z = x$ (위치만).

$$\boldsymbol{F} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}, \quad \boldsymbol{H} = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

$$\boldsymbol{Q} = \begin{pmatrix} q_x & 0 \\ 0 & q_v \end{pmatrix}, \quad \boldsymbol{R} = r$$

- 측정은 위치만 — 그러나 KF는 속도까지 추정한다
- 그 비밀은 cross-covariance. 시간이 흐르면 $\boldsymbol{P}$의 비대각이 자란다

왜 위치만 측정해도 속도가 추정되는가

- predict step에서 $\boldsymbol{P}$가 $\boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^\top$로 갱신
- $\boldsymbol{F}$의 Δt 항이 대각 외에 정보를 심는다
- $\boldsymbol{P}_{12}$가 0이 아니게 되면, 위치 측정이 속도 추정에까지 영향

왜 위치만 측정해도 속도가 추정되는가

- predict step에서 $\boldsymbol{P}$가 $\boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^\top$로 갱신
- $\boldsymbol{F}$의 Δt 항이 대각 외에 정보를 심는다
- $\boldsymbol{P}_{12}$가 0이 아니게 되면, 위치 측정이 속도 추정에까지 영향

한 줄 요약
KF는 측정 안 한 변수도 모델에 의해 다른 변수와 묶이면 함께 좁아진다.

bias 자동 추정의 원리도 정확히 이것 (5회차).

1D 위치+속도 KF — Python 30줄 (1/2)


```python
import numpy as np

dt = 0.1
F = np.array([[1, dt], [0, 1]])
H = np.array([[1, 0]])
Q = np.diag([1e-3, 1e-2])
R = np.array([[0.5]])

x = np.array([[0.0], [0.0]])
P = np.eye(2) * 100.0       # 초기분산크게

def predict(x, P):
    x = F @ x
    P = F @ P @ F.T + Q
    return x, P
```

1D 위치+속도 KF — Python 30줄 (2/2)


```
def update(x, P, z):
    y = z - H @ x
    S = H @ P @ H.T + R
    K = P @ H.T @ np.linalg.inv(S)
    x = x + K @ y
    P = (np.eye(2) - K @ H) @ P
    return x, P

# 진실: x(t) = 0.5*t 등속 ( 0.5)
true_x = 0.0
for k in range(100):
    true_x += 0.5 * dt
    x, P = predict(x, P)
    if k % 5 == 0:
        z = np.array([[true_x + np.random.randn()*0.7]])
        x, P = update(x, P, z)
print(x.ravel(), np.diag(P))
```

돌려보면 — 무엇을 관찰하나

- 처음 몇 step: 분산이 빨리 떨어짐 (초기 큰 분산이 ”흡수”)
- 정상 상태: $\boldsymbol{P}$가 일정한 모양으로 수렴 (dt, Q, R에 의해 결정)
- 측정 끊으면 (e.g. $k\%50$): 분산이 다시 커지고 평균은 dynamics만으로 외삽

돌려보면 — 무엇을 관찰하나

- 처음 몇 step: 분산이 빨리 떨어짐 (초기 큰 분산이 ”흡수”)
- 정상 상태: $\boldsymbol{P}$가 일정한 모양으로 수렴 (dt, Q, R에 의해 결정)
- 측정 끊으면 (e.g. $k\%50$): 분산이 다시 커지고 평균은 dynamics만으로 외삽

과제 사고 실험
위 코드에서 측정 빈도를 $k\%5 \rightarrow k\%20$로 바꾸면 속도 추정 분산이 어떻게 되나?
GPS 끊긴 터널을 흉내내본 셈.

KF는 최적 추정량

- 가정 4개(선형, 가우시안, 잡음 독립, 모델 정확) 아래에서
- KF는 **Minimum Mean Square Error** 추정량
- 즉 어떤 다른 알고리즘도 평균 제곱 오차에서 KF를 이길 수 없다

KF는 최적 추정량

- 가정 4개(선형, 가우시안, 잡음 독립, 모델 정확) 아래에서
- KF는 **Minimum Mean Square Error** 추정량
- 즉 어떤 다른 알고리즘도 평균 제곱 오차에서 KF를 이길 수 없다

단서 조항
가정이 깨지면 (비선형, 비가우시안, 모델 오차) 더 이상 최적이 아님.
그래서 EKF, UKF, particle filter가 등장 (4회차부터).

KF의 두 가지 등가 시각

1. **Bayesian:** prior $\times$ likelihood = posterior
2. **Frequentist (least squares):** 잔차 제곱합을 inverse covariance로 가중하여 최소화

KF의 두 가지 등가 시각

1. **Bayesian:** prior $\times$ likelihood = posterior
2. **Frequentist (least squares):** 잔차 제곱합을 inverse covariance로 가중하여 최소화

둘이 같은 답을 내는 이유
가우시안의 log를 취하면 이차 형식 → 최소화 = 평균 찾기.
그래서 두 시각이 정확히 일치하는 유일한 분포가 가우시안.

Information form (잠깐만)

P^{-1}을 **information matrix** Λ, $\xi = \Lambda\hat{x}$를 **information vector**.

Information form (잠깐만)

P^{-1}을 **information matrix** $\Lambda, \xi = \Lambda\hat{x}$를 **information vector**. Update가 덧셈으로 표현됨:

$$\Lambda_{k|k} = \Lambda_{k|k-1} + \boldsymbol{H}^\top \boldsymbol{R}^{-1} \boldsymbol{H}$$

$$\boldsymbol{\xi}_{k|k} = \boldsymbol{\xi}_{k|k-1} + \boldsymbol{H}^\top \boldsymbol{R}^{-1} \boldsymbol{z}_k$$

Information form (잠깐만)

P^{-1}을 **information matrix** $\Lambda, \xi = \Lambda\hat{x}$를 **information vector**. Update가 덧셈으로 표현됨:

$$\Lambda_{k|k} = \Lambda_{k|k-1} + \boldsymbol{H}^\top \boldsymbol{R}^{-1} \boldsymbol{H}$$

$$\boldsymbol{\xi}_{k|k} = \boldsymbol{\xi}_{k|k-1} + \boldsymbol{H}^\top \boldsymbol{R}^{-1} \boldsymbol{z}_k$$

- 다중 센서 결합이 자연스러움 (그냥 더한다)
- 단점: predict가 복잡해짐
- 분산형/SLAM에서 자주 등장 — factor graph의 정신적 뿌리

KF 식의 감각 — 어디서 나온 게인지

$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{H}^\top \boldsymbol{S}^{-1}$ 한 번 더 쪼개기:

- $\boldsymbol{P}$ = state 불확실성 (현재)
- $\boldsymbol{H}$ = state $\rightarrow$ 측정 변환
- $\boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^\top$ = 측정 공간에서 본 state 불확실성
- $\boldsymbol{R}$ = 측정 잡음 분산
- $\boldsymbol{S} = \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^\top + \boldsymbol{R}$ = innovation의 전체 불확실성
- $\boldsymbol{S}^{-1}$ = 정밀도

KF 식의 감각 — 어디서 나온 게인지

$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{H}^\top \boldsymbol{S}^{-1}$ 한 번 더 쪼개기:

- $\boldsymbol{P}$ = state 불확실성 (현재)
- $\boldsymbol{H}$ = state $\to$ 측정 변환
- $\boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^\top$ = 측정 공간에서 본 state 불확실성
- $\boldsymbol{R}$ = 측정 잡음 분산
- $\boldsymbol{S} = \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^\top + \boldsymbol{R}$ = innovation의 전체 불확실성
- $\boldsymbol{S}^{-1}$ = 정밀도

비유
"내가 얼마나 헷갈리는지" $\times$ "측정이 얼마나 신뢰할 만한지의 역수" = 보정의 강도.

비선형으로 가기 전 한 슬라이드 정리

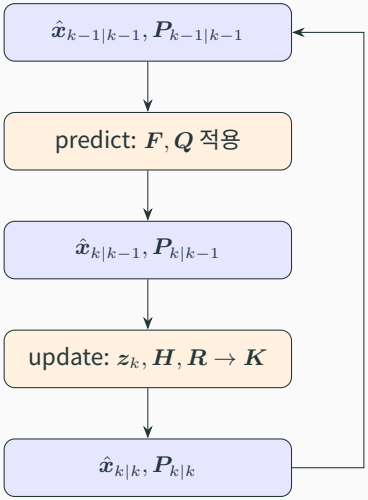
- 5줄 KF는 4가정 위에서 **유일하고 최적**
- IMU+회전 문제는 quaternion (비선형 다양체) → 가정이 깨진다
- **핵심 해법: ESKF** — 오차 상태를 $\mathbb{R}^{15}$에서 선형 KF로 추정
- 명목 상태는 비선형 운동방정식으로 전파 → 두 층의 역할 분리

비선형으로 가기 전 한 슬라이드 정리

- 5줄 KF는 4가정 위에서 **유일하고 최적**
- IMU+회전 문제는 quaternion (비선형 다양체) → 가정이 깨진다
- **핵심 해법: ESKF** — 오차 상태를 $\mathbb{R}^{15}$에서 선형 KF로 추정
- 명목 상태는 비선형 운동방정식으로 전파 → 두 층의 역할 분리

다음 회차 미리보기
4회차: 오차 상태의 정의 · F 행렬 유도 · Inject & Reset
(EKF/UKF는 역사적 맥락으로만 간단히 언급)

이번 회차의 한 슬라이드 다이어그램



이번 회차 핵심 다섯 개

1. KF = Bayes filter + 4가정 (선형, 선형, 가우시안, 잡음 독립)
2. Predict: 평균은 옮기고, 공분산은 회전+늘림+$\boldsymbol{Q}$
3. Update: $\boldsymbol{K} = \boldsymbol{P}\boldsymbol{H}^{\top}\boldsymbol{S}^{-1}$ — cross-cov $\times$ 측정 정밀도
4. 측정 안 한 변수도 모델에 의해 묶이면 함께 좁아진다
5. 5줄을 유도할 수 있어야 본인 것

과제 3 (다음 시간 전)

1. 슬라이드 7의 공분산 식을 종이에 다시 유도 (잡음 독립성 어디서 쓰였는지 표시)
2. 슬라이드 12의 $\partial \mathrm{tr}\boldsymbol{P}/\partial \boldsymbol{K} = 0$ 풀어 $\boldsymbol{K}$ 도출 (행렬 미분 공식: $\partial \mathrm{tr}(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B})/\partial \boldsymbol{X} = \boldsymbol{A}^\top \boldsymbol{B}^\top$)
3. 23-24번 코드에서 측정 빈도 5/20/100을 비교, σ_x, σ_v의 시간 변화를 그래프로
4. (선택) Joseph form vs short form을 일부러 $\boldsymbol{R}$을 매우 작게 (1e-8) 두고 비교 — 어느 쪽이 더 안정한가?

다음 시간 예고 — 4회차: ESKF 핵심

Error-State Kalman Filter — 명목 상태 + 오차 상태의 두 층

- 오차 상태 $\delta \boldsymbol{x} \in \mathbb{R}^{15}$ 정의: $\delta \boldsymbol{p}, \delta \boldsymbol{v}, \delta \boldsymbol{\phi}, \delta \boldsymbol{b}_g, \delta \boldsymbol{b}_a$
- F 행렬 유도 — OpenVINS `Propagator.cpp` 실제 코드로 확인
- Inject & Reset: 업데이트 후 $\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}} \oplus \delta\hat{\boldsymbol{x}}$, $\delta \boldsymbol{x} \leftarrow 0$
- 왜 quaternion 부호 문제가 사라지는가

미리 생각해볼 것
$\delta \boldsymbol{\phi} \in \mathbb{R}^3$이면 KF가 그냥 작동한다.
quaternion $\boldsymbol{q} \in \mathbb{R}^4$ (단위 구속)을 직접 KF 상태로 두면 어떤 문제가 생기는가?

질문 환영

다음 시간엔 오차 상태라는 우아한 해법을 만난다

4회차: ESKF 핵심

KF = 4가정 아래 최적. 회전은 SO(3) 다양체 — 유클리드 KF 직접 적용 불가.

오늘 질문: **회전이 있으면 어떻게 KF를 쓰는가?**

- 필터(ESKF/iEKF) 기반: OpenVINS, MSCKF, ROVIO, FAST-LIO2, FAST-LIVO2
- 인접 영역 (factor graph + preintegration): VINS-Mono, ORB-SLAM3 VI, LIO-SAM, Kalibr (batch)
- 공통 회전 처리: **Error-State** 표현 + SO(3) 매니폴드
- 핵심 아이디어: 회전을 상태로 넣지 말고, 회전 주변 오차를 상태로 넣는다

왜 EKF로는 부족한가

EKF 접근: quaternion $\boldsymbol{q} \in \mathbb{R}^4$를 상태로 넣고, 공분산 $\boldsymbol{P} \in \mathbb{R}^{n \times n}$을 n에 포함

세 가지 문제

1. **단위 구속**: $\|\boldsymbol{q}\| = 1$ 이어야 하는데 KF 업데이트 후 깨짐 → 매 step 정규화
2. **부호 이중성**: $\boldsymbol{q}$와 $-\boldsymbol{q}$는 같은 회전 → 공분산이 의미를 잃음
3. **과도추정**: 4D 공분산이 3D 자유도를 표현 → 행렬이 항상 rank-deficient

왜 EKF로는 부족한가

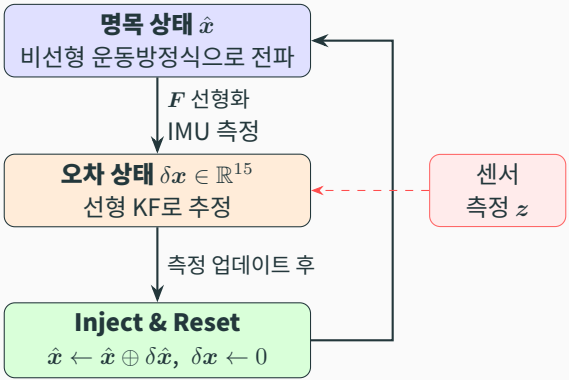
EKF 접근: quaternion $\boldsymbol{q} \in \mathbb{R}^4$를 상태로 넣고, 공분산 $\boldsymbol{P} \in \mathbb{R}^{n \times n}$을 n에 포함

세 가지 문제

1. **단위 구속**: $\|\boldsymbol{q}\| = 1$ 이어야 하는데 KF 업데이트 후 깨짐 → 매 step 정규화
2. **부호 이중성**: $\boldsymbol{q}$와 $-\boldsymbol{q}$는 같은 회전 → 공분산이 의미를 잃음
3. **과도추정**: 4D 공분산이 3D 자유도를 표현 → 행렬이 항상 rank-deficient

해결책
회전 오차 $\delta\boldsymbol{\phi} \in \mathbb{R}^3$만 KF 상태로 쓴다 — 제약 없음, rank-deficiency 없음.

ESKF의 두 층 구조


명목 상태 $\hat{x}$
비선형 운동방정식으로 전파
F 선형화
IMU 측정
오차 상태 $\delta x \in \mathbb{R}^{15}$
선형 KF로 추정
센서
측정 z
측정 업데이트 후
Inject & Reset
$\hat{x} \leftarrow \hat{x} \oplus \delta\hat{x},\ \delta x \leftarrow 0$

오차 상태 $\delta \boldsymbol{x} \in \mathbb{R}^{15}$

변수	명목	오차
위치	$\hat{\boldsymbol{p}} \in \mathbb{R}^3$	$\delta \boldsymbol{p} = \boldsymbol{p} - \hat{\boldsymbol{p}}$
속도	$\hat{\boldsymbol{v}} \in \mathbb{R}^3$	$\delta \boldsymbol{v} = \boldsymbol{v} - \hat{\boldsymbol{v}}$
회전	$\hat{\boldsymbol{R}} \in \mathrm{SO}(3)$	$\delta \boldsymbol{\phi} \in \mathbb{R}^3, \boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta \boldsymbol{\phi})$
자이로 바이어스	$\hat{\boldsymbol{b}}_g \in \mathbb{R}^3$	$\delta \boldsymbol{b}_g = \boldsymbol{b}_g - \hat{\boldsymbol{b}}_g$
가속도 바이어스	$\hat{\boldsymbol{b}}_a \in \mathbb{R}^3$	$\delta \boldsymbol{b}_a = \boldsymbol{b}_a - \hat{\boldsymbol{b}}_a$

오차 상태 $\delta \boldsymbol{x} \in \mathbb{R}^{15}$

변수	명목	오차
위치	$\hat{\boldsymbol{p}} \in \mathbb{R}^3$	$\delta \boldsymbol{p} = \boldsymbol{p} - \hat{\boldsymbol{p}}$
속도	$\hat{\boldsymbol{v}} \in \mathbb{R}^3$	$\delta \boldsymbol{v} = \boldsymbol{v} - \hat{\boldsymbol{v}}$
회전	$\hat{\boldsymbol{R}} \in \mathrm{SO}(3)$	$\delta \boldsymbol{\phi} \in \mathbb{R}^3, \boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta \boldsymbol{\phi})$
자이로 바이어스	$\hat{\boldsymbol{b}}_g \in \mathbb{R}^3$	$\delta \boldsymbol{b}_g = \boldsymbol{b}_g - \hat{\boldsymbol{b}}_g$
가속도 바이어스	$\hat{\boldsymbol{b}}_a \in \mathbb{R}^3$	$\delta \boldsymbol{b}_a = \boldsymbol{b}_a - \hat{\boldsymbol{b}}_a$

핵심

$\delta \boldsymbol{\phi} \in \mathbb{R}^3$ — 제약 없음. KF는 $\mathbb{R}$에서만 작동 가능.

15D 오차 벡터 구조 (OpenVINS 순서)

$$\delta \boldsymbol{x} = \begin{bmatrix} \delta \boldsymbol{\phi} \\ \delta \boldsymbol{p} \\ \delta \boldsymbol{v} \\ \delta \boldsymbol{b}_g \\ \delta \boldsymbol{b}_a \end{bmatrix} \in \mathbb{R}^{15}$$

상태 순서는 구현마다 다를 수 있음. OpenVINS: $[\delta\boldsymbol{\phi},\, \delta\boldsymbol{p},\, \delta\boldsymbol{v},\, \delta\boldsymbol{b}_g,\, \delta\boldsymbol{b}_a]^\top$.

True = Nominal $\oplus$ Error

각 성분의 합성 규칙 (**composition rule**):

$$
\begin{aligned}
\boldsymbol{p} &= \hat{\boldsymbol{p}} + \delta\boldsymbol{p} \\
\boldsymbol{v} &= \hat{\boldsymbol{v}} + \delta\boldsymbol{v} \\
\boldsymbol{R} &= \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi}) \quad \leftarrow \text{오른쪽 섭동(right perturbation)} \\
\boldsymbol{b}_g &= \hat{\boldsymbol{b}}_g + \delta\boldsymbol{b}_g \\
\boldsymbol{b}_a &= \hat{\boldsymbol{b}}_a + \delta\boldsymbol{b}_a
\end{aligned}
$$

True = Nominal $\oplus$ Error

각 성분의 합성 규칙 (**composition rule**):

$$
\begin{aligned}
\boldsymbol{p} &= \hat{\boldsymbol{p}} + \delta\boldsymbol{p} \\
\boldsymbol{v} &= \hat{\boldsymbol{v}} + \delta\boldsymbol{v} \\
\boldsymbol{R} &= \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi}) \quad \leftarrow \text{오른쪽 섭동(right perturbation)} \\
\boldsymbol{b}_g &= \hat{\boldsymbol{b}}_g + \delta\boldsymbol{b}_g \\
\boldsymbol{b}_a &= \hat{\boldsymbol{b}}_a + \delta\boldsymbol{b}_a
\end{aligned}
$$

왜 오른쪽 섭동인가
$\boldsymbol{R} = \mathrm{Exp}(\delta\boldsymbol{\phi})\,\hat{\boldsymbol{R}}$ (왼쪽)도 가능하지만, 오른쪽이 Body frame 오차로 더 자연스럽다. OpenVINS, VINS-Mono 모두 오른쪽 섭동 사용.

명목 상태 전파 — IMU 기반

IMU 측정값: $\boldsymbol{\omega}_m = \boldsymbol{\omega} + \boldsymbol{b}_g + \boldsymbol{n}_g,\ \boldsymbol{a}_m = \boldsymbol{a} + \boldsymbol{b}_a + \boldsymbol{n}_a$

편의 표기: $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\boldsymbol{b}}_g,\ \tilde{\boldsymbol{a}} = \boldsymbol{a}_m - \hat{\boldsymbol{b}}_a$ (바이어스 보정 측정값)

연속시간 명목 상태 미분방정식:

$$
\begin{aligned}
\dot{\hat{\boldsymbol{p}}} &= \hat{\boldsymbol{v}} \\
\dot{\hat{\boldsymbol{v}}} &= \hat{\boldsymbol{R}}\,\tilde{\boldsymbol{a}} + \boldsymbol{g} \\
\dot{\hat{\boldsymbol{R}}} &= \hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{\omega}}]_\times \\
\dot{\hat{\boldsymbol{b}}}_g &= 0 \quad \text{(임의보행 모델)} \\
\dot{\hat{\boldsymbol{b}}}_a &= 0
\end{aligned}
$$

명목 상태 전파 — IMU 기반

IMU 측정값: $\boldsymbol{\omega}_m = \boldsymbol{\omega} + \boldsymbol{b}_g + \boldsymbol{n}_g,\ \boldsymbol{a}_m = \boldsymbol{a} + \boldsymbol{b}_a + \boldsymbol{n}_a$

편의 표기: $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\boldsymbol{b}}_g,\ \tilde{\boldsymbol{a}} = \boldsymbol{a}_m - \hat{\boldsymbol{b}}_a$ (바이어스 보정 측정값)

연속시간 명목 상태 미분방정식:

$$
\begin{aligned}
\dot{\hat{\boldsymbol{p}}} &= \hat{\boldsymbol{v}} \\
\dot{\hat{\boldsymbol{v}}} &= \hat{\boldsymbol{R}}\,\tilde{\boldsymbol{a}} + \boldsymbol{g} \\
\dot{\hat{\boldsymbol{R}}} &= \hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{\omega}}]_\times \\
\dot{\hat{\boldsymbol{b}}}_g &= 0 \quad \text{(임의보행 모델)} \\
\dot{\hat{\boldsymbol{b}}}_a &= 0
\end{aligned}
$$

이산화: $\hat{\boldsymbol{R}}_{k+1} = \hat{\boldsymbol{R}}_k\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}\,\Delta t)$ (1차 근사)

오차 상태 선형화 — F 행렬

오차 상태 동역학 $\dot{\delta \boldsymbol{x}} \approx \boldsymbol{F}\,\delta \boldsymbol{x} + \boldsymbol{G}\,\boldsymbol{n}$에서

연속시간 F 행렬 (15×15):

$$\boldsymbol{F} = \begin{pmatrix} -[\tilde{\boldsymbol{\omega}}]_\times & 0 & 0 & -\boldsymbol{I} & 0 \\ 0 & 0 & \boldsymbol{I} & 0 & 0 \\ -\hat{\boldsymbol{R}}[\tilde{\boldsymbol{a}}]_\times & 0 & 0 & 0 & -\hat{\boldsymbol{R}} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} \leftarrow \delta\boldsymbol{\phi} \\ \leftarrow \delta\boldsymbol{p} \\ \leftarrow \delta\boldsymbol{v} \\ \leftarrow \delta\boldsymbol{b}_g \\ \leftarrow \delta\boldsymbol{b}_a \end{matrix}$$

오차 상태 선형화 — F 행렬

오차 상태 동역학 $\dot{\delta \boldsymbol{x}} \approx \boldsymbol{F}\,\delta \boldsymbol{x} + \boldsymbol{G}\,\boldsymbol{n}$에서

연속시간 F 행렬 (15×15):

$$\boldsymbol{F} = \begin{pmatrix} -[\tilde{\boldsymbol{\omega}}]_\times & 0 & 0 & -\boldsymbol{I} & 0 \\ 0 & 0 & \boldsymbol{I} & 0 & 0 \\ -\hat{\boldsymbol{R}}[\tilde{\boldsymbol{a}}]_\times & 0 & 0 & 0 & -\hat{\boldsymbol{R}} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} \leftarrow \delta\boldsymbol{\phi} \\ \leftarrow \delta\boldsymbol{p} \\ \leftarrow \delta\boldsymbol{v} \\ \leftarrow \delta\boldsymbol{b}_g \\ \leftarrow \delta\boldsymbol{b}_a \end{matrix}$$

이산화: $\boldsymbol{F}_d \approx \boldsymbol{I} + \boldsymbol{F}\Delta t + \frac{1}{2}\boldsymbol{F}^2\Delta t^2$

OpenVINS의 Φ_d — 실제 코드 (변수명은 F이나 내용은 이산 전이행렬)

ov_msckf/src/update/Propagator.cpp — compute_F_and_G_analytic()

주의: OpenVINS는 변수명을 F로 쓰지만, 실제 내용은 $\boldsymbol{\Phi}_d \approx \boldsymbol{I} + \boldsymbol{F}\Delta t + \frac{1}{2}\boldsymbol{F}^2\Delta t^2$ — 즉 이산화 후의 전이행렬.

앞 슬라이드의 연속 $\boldsymbol{F}$와 직접 비교하려면 $\boldsymbol{I} + \boldsymbol{F}\Delta t$를 머릿속에서 더해야 한다.

```
// State order: [theta(3), p(3), v(3), bg(3), ba(3)]
// Phi_thetatheta  ~ I - [omega_tilde]x * dt 코드에서는  ( 분리해서쌓음 )
F.block(0,0,3,3) = -skew(w_hat);
// Phi_thetabg     = -Jr(omega*dt)*dt 소각 (: ~ -I*dt)
F.block(0,9,3,3) = -Xi_1.block(0,0,3,3);
// Phi_vtheta      = -R*[a_tilde]x*dt
F.block(6,0,3,3) = -R_Gtoi * skew(a_hat);
// Phi_vba         = -R*dt
F.block(6,12,3,3) = -R_Gtoi;
// Phi_pv          = I*dt
F.block(3,6,3,3) = Eigen::Matrix3d::Identity();
// Phi_ptheta      = -0.5*R*[a_tilde]x*dt^2 이산화 ( 차2 보정항—연속에는   F 없음)
F.block(3,0,3,3) = -0.5 * dt * R_Gtoi * skew(a_hat);
```

출처: github.com/rpng/open_vins (GPLv3 라이선스). $\boldsymbol{R}_{\text{Gtoi}}$ = $\hat{\boldsymbol{R}}$ (World→IMU), a_hat = $\tilde{\boldsymbol{a}}$, w_hat = $\tilde{\boldsymbol{\omega}}$.

오차 상태 Predict step

1. **명목 상태 전파** (비선형, Runge-Kutta 또는 1차 적분)

$$\hat{\boldsymbol{x}}_{k+1} = f(\hat{\boldsymbol{x}}_k,\, \boldsymbol{u}_k)$$

2. **오차 공분산 전파** (선형 KF 그대로)

$$\boldsymbol{P}_{k+1|k} = \boldsymbol{F}_d\, \boldsymbol{P}_{k|k}\, \boldsymbol{F}_d^{\top} + \boldsymbol{G}_d\, \boldsymbol{Q}\, \boldsymbol{G}_d^{\top}$$

오차 상태 Predict step

1. **명목 상태 전파** (비선형, Runge-Kutta 또는 1차 적분)

$$\hat{\boldsymbol{x}}_{k+1} = f(\hat{\boldsymbol{x}}_k,\, \boldsymbol{u}_k)$$

2. **오차 공분산 전파** (선형 KF 그대로)

$$\boldsymbol{P}_{k+1|k} = \boldsymbol{F}_d\, \boldsymbol{P}_{k|k}\, \boldsymbol{F}_d^\top + \boldsymbol{G}_d\, \boldsymbol{Q}\, \boldsymbol{G}_d^\top$$

핵심 분리
명목 상태는 비선형 정확한 역학으로, 오차 상태는 간단한 선형 KF로.
두 층이 역할을 나눠 갖는다 — 이것이 ESKF의 elegance.

오차 상태 Update step

측정 모델: $\boldsymbol{z} = h(\hat{\boldsymbol{x}}) + \boldsymbol{H}\,\delta\boldsymbol{x} + \boldsymbol{v}$

H는 명목 상태에서 평가한 Jacobian:

$$\boldsymbol{H} = \left.\frac{\partial h}{\partial \delta \boldsymbol{x}}\right|_{\delta \boldsymbol{x}=0}$$

오차 상태 Update step

측정 모델: $\boldsymbol{z} = h(\hat{\boldsymbol{x}}) + \boldsymbol{H}\,\delta\boldsymbol{x} + \boldsymbol{v}$

H는 명목 상태에서 평가한 Jacobian:

$$\boldsymbol{H} = \left.\frac{\partial h}{\partial \delta \boldsymbol{x}}\right|_{\delta \boldsymbol{x}=0}$$

Innovation & Kalman gain (표준 KF):

$$\begin{aligned} \boldsymbol{r} &= \boldsymbol{z} - h(\hat{\boldsymbol{x}}) \quad \text{(residual)} \\ \boldsymbol{S} &= \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^{\top} + \boldsymbol{R} \\ \boldsymbol{K} &= \boldsymbol{P}\boldsymbol{H}^{\top}\boldsymbol{S}^{-1} \end{aligned}$$

오차 상태 Update step

측정 모델: $\boldsymbol{z} = h(\hat{\boldsymbol{x}}) + \boldsymbol{H}\,\delta\boldsymbol{x} + \boldsymbol{v}$

H는 명목 상태에서 평가한 Jacobian:

$$\boldsymbol{H} = \left.\frac{\partial h}{\partial \delta \boldsymbol{x}}\right|_{\delta \boldsymbol{x}=0}$$

Innovation & Kalman gain (표준 KF):

$$\begin{aligned}
\boldsymbol{r} &= \boldsymbol{z} - h(\hat{\boldsymbol{x}}) \quad \text{(residual)} \\
\boldsymbol{S} &= \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^{\top} + \boldsymbol{R} \\
\boldsymbol{K} &= \boldsymbol{P}\boldsymbol{H}^{\top}\boldsymbol{S}^{-1}
\end{aligned}$$

오차 상태 업데이트 + Joseph form:

$$\begin{aligned}
\delta\hat{\boldsymbol{x}} &= \boldsymbol{K}\boldsymbol{r} \\
\boldsymbol{P}^{+} &= (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})^{\top} + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^{\top} \quad \leftarrow \text{항상}
\end{aligned}$$

Inject & Reset — 가장 중요한 단계

업데이트 후 오차 추정치 $\delta\hat{\boldsymbol{x}}$를 명목 상태에 **주입(inject)**하고 오차를 **초기화(reset)**:

$$
\begin{aligned}
\hat{\boldsymbol{p}} &\leftarrow \hat{\boldsymbol{p}} + \delta\hat{\boldsymbol{p}} \\
\hat{\boldsymbol{v}} &\leftarrow \hat{\boldsymbol{v}} + \delta\hat{\boldsymbol{v}} \\
\hat{\boldsymbol{R}} &\leftarrow \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\hat{\boldsymbol{\phi}}) \quad \leftarrow \text{SO(3) 가환 없음} \\
\hat{\boldsymbol{b}}_g &\leftarrow \hat{\boldsymbol{b}}_g + \delta\hat{\boldsymbol{b}}_g \\
\hat{\boldsymbol{b}}_a &\leftarrow \hat{\boldsymbol{b}}_a + \delta\hat{\boldsymbol{b}}_a
\end{aligned}
$$

Inject & Reset — 가장 중요한 단계

업데이트 후 오차 추정치 $\delta\hat{\boldsymbol{x}}$를 명목 상태에 **주입(inject)**하고 오차를 **초기화(reset)**:

$$
\begin{aligned}
\hat{\boldsymbol{p}} &\leftarrow \hat{\boldsymbol{p}} + \delta\hat{\boldsymbol{p}} \\
\hat{\boldsymbol{v}} &\leftarrow \hat{\boldsymbol{v}} + \delta\hat{\boldsymbol{v}} \\
\hat{\boldsymbol{R}} &\leftarrow \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\hat{\boldsymbol{\phi}}) \quad \leftarrow \text{SO(3) 가환 없음} \\
\hat{\boldsymbol{b}}_g &\leftarrow \hat{\boldsymbol{b}}_g + \delta\hat{\boldsymbol{b}}_g \\
\hat{\boldsymbol{b}}_a &\leftarrow \hat{\boldsymbol{b}}_a + \delta\hat{\boldsymbol{b}}_a
\end{aligned}
$$

그런 다음:

$$
\delta\boldsymbol{x} \leftarrow 0, \quad \boldsymbol{P} \leftarrow \boldsymbol{G}_{\text{reset}}\,\boldsymbol{P}^{+}\,\boldsymbol{G}_{\text{reset}}^{\top}
$$

($\boldsymbol{G}_{\text{reset}} \approx \boldsymbol{I}$ for small $\delta\hat{\boldsymbol{\phi}}$)

왜 quaternion 부호 문제가 사라지는가

EKF (quaternion 직접 추정)

- 상태: $\boldsymbol{q} \in \mathbb{R}^4$
- $\boldsymbol{q}$와 $-\boldsymbol{q}$가 같은 회전
- KF 업데이트가 두 모드 사이를 점프할 수 있음
- 공분산이 음반정치로 발산 가능

ESKF (오차 상태)

- KF 상태: $\delta\boldsymbol{\phi} \in \mathbb{R}^3$
- 부호 이중성 없음 — $\mathbb{R}^3$는 평탄
- 명목 $\hat{\boldsymbol{R}}$이 부호 정보를 들고 있음
- 오차 $\delta\boldsymbol{\phi}$은 항상 “0 근처”

왜 quaternion 부호 문제가 사라지는가

EKF (quaternion 직접 추정)

- 상태: $\boldsymbol{q} \in \mathbb{R}^4$
- $\boldsymbol{q}$와 $-\boldsymbol{q}$가 같은 회전
- KF 업데이트가 두 모드 사이를 점프할 수 있음
- 공분산이 음반정치로 발산 가능

ESKF (오차 상태)

- KF 상태: $\delta\boldsymbol{\phi} \in \mathbb{R}^3$
- 부호 이중성 없음 — $\mathbb{R}^3$는 평탄
- 명목 $\hat{\boldsymbol{R}}$이 부호 정보를 들고 있음
- 오차 $\delta\boldsymbol{\phi}$은 항상 "0 근처"

핵심 통찰
ESKF는 quaternion을 쓰지 않는 것이 아니라, 명목 상태에 저장하되 KF가 건드리는 것은 오직 $\mathbb{R}^3$ 오차뿐. 그래서 모든 문제가 사라진다.

ESKF 전체 흐름 — 한 싸이클

1. **IMU 수신** ($\boldsymbol{\omega}_m, \boldsymbol{a}_m$)
2. **명목 전파**: $\hat{\boldsymbol{p}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{R}}, \hat{\boldsymbol{b}}_g, \hat{\boldsymbol{b}}_a$ 업데이트 (비선형)
3. **공분산 전파**: $\boldsymbol{P} \leftarrow \boldsymbol{F}_d \boldsymbol{P} \boldsymbol{F}_d^\top + \boldsymbol{G}_d \boldsymbol{Q} \boldsymbol{G}_d^\top$
4. **센서 수신** (GPS / 카메라 / 휠 등)
5. **residual 계산**: $\boldsymbol{r} = \boldsymbol{z} - h(\hat{\boldsymbol{x}})$
6. **칼만 게인**: $\boldsymbol{K} = \boldsymbol{P}\boldsymbol{H}^\top (\boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^\top + \boldsymbol{R})^{-1}$
7. **오차 업데이트**: $\delta\hat{\boldsymbol{x}} = \boldsymbol{K}\boldsymbol{r}$
8. **공분산 업데이트**: Joseph form
9. **Inject & Reset**: $\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}} \oplus \delta\hat{\boldsymbol{x}}$, $\delta\boldsymbol{x} \leftarrow 0$
10. 다음 IMU가 올 때까지 대기 → 1로 돌아감

VINS-Mono midpoint 적분 — F 행렬 (참고)

estimator/src/factor/integration_base.h

```
// 15x15 state: [p(3), q(3 angle), v(3), ba(3), bg(3)]
Matrix<double,15,15> F = Matrix<double,15,15>::Zero();
F.block<3,3>(0,3)   = -0.5 * delta_q.toRotationMatrix() * R_a_0_x * dt;
F.block<3,3>(0,6)   =  0.5 * dt * Matrix3d::Identity();
F.block<3,3>(0,9)   = -0.25 * (delta_q.toRotationMatrix()) * dt * dt;
F.block<3,3>(3,3)   = -R_w_x * dt;
F.block<3,3>(3,12)  = -1.0 * MatrixXd::Identity(3,3) * dt;
F.block<3,3>(6,3)   = -delta_q.toRotationMatrix() * R_a_0_x * dt;
F.block<3,3>(6,6)   =  MatrixXd::Identity(3,3);
F.block<3,3>(6,9)   = -delta_q.toRotationMatrix() * dt;
```

출처: github.com/HKUST-Aerial-Robotics/VINS-Mono (GPL-3.0)

상태 순서만 다를 뿐 구조는 동일. R_a_0_x는 $[\tilde{\boldsymbol{a}}]_\times$, R_w_x는 $[\tilde{\boldsymbol{\omega}}]_\times$에 해당.

EKF / UKF / ESKF 비교

	EKF	UKF	ESKF
회전 표현	quaternion (4D)	quaternion (4D)	$\delta\phi$ (3D)
부호 이중성	문제	문제	**없음**
rank-deficiency	있음	있음	**없음**
Jacobian 필요	있음	**없음**	있음 (F, H)
계산 복잡도	$O(n^2)$	$O(n^3)$ sigma pts	$O(n^2)$
산업 채택	중간	소수	**압도적 다수**
대표 시스템	초기 SLAM	일부 AHRS	OpenVINS, MSCKF, ROVIO, FAST-LIO2, FAST-LIVO2

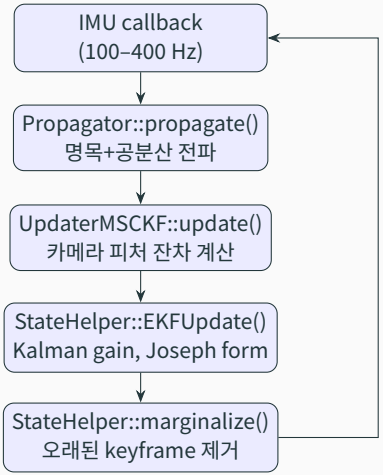
EKF / UKF / ESKF 비교

	EKF	UKF	ESKF
회전 표현	quaternion (4D)	quaternion (4D)	$\delta\phi$ (3D)
부호 이중성	문제	문제	**없음**
rank-deficiency	있음	있음	**없음**
Jacobian 필요	있음	**없음**	있음 (F, H)
계산 복잡도	$O(n^2)$	$O(n^3)$ sigma pts	$O(n^2)$
산업 채택	중간	소수	**압도적 다수**
대표 시스템	초기 SLAM	일부 AHRS	OpenVINS, MSCKF, ROVIO, FAST-LIO2, FAST-LIVO2

결론

UKF는 Jacobian 계산이 필요 없다는 장점이 있지만, sigma point 전파가 SO(3)에서 여전히 까다롭다. ESKF가 이론적으로 더 깔끔하고 실용적으로 더 빠르다.

실제 코드에서 ESKF 찾기 — OpenVINS 구조



State.h
JPLQuat
IMUData

ESKF의 “오차는 항상 0 근처” 가정

ESKF 선형화의 유효성:

- F 행렬은 $\delta \boldsymbol{x} = 0$에서의 1차 Taylor 근사
- 오차 $\delta \boldsymbol{x}$이 작을수록 근사가 정확

ESKF의 “오차는 항상 0 근처” 가정

ESKF 선형화의 유효성:

- F 행렬은 $\delta \boldsymbol{x} = 0$에서의 1차 Taylor 근사
- 오차 $\delta \boldsymbol{x}$이 작을수록 근사가 정확

이것이 보장되는 이유
Inject & Reset이 매 측정마다 $\delta \boldsymbol{x} \leftarrow 0$으로 초기화.
명목 상태가 실제 상태를 항상 따라가기 때문에 오차는 작다.

ESKF의 “오차는 항상 0 근처” 가정

ESKF 선형화의 유효성:

- F 행렬은 $\delta \boldsymbol{x} = 0$에서의 1차 Taylor 근사
- 오차 $\delta \boldsymbol{x}$이 작을수록 근사가 정확

이것이 보장되는 이유
Inject & Reset이 매 측정마다 $\delta \boldsymbol{x} \leftarrow 0$으로 초기화.
명목 상태가 실제 상태를 항상 따라가기 때문에 오차는 작다.

무너지는 경우
센서가 오랫동안 끊기면 명목 상태가 실제와 크게 벌어지고,
Reset 없이 오차가 커지면 선형화 오차도 커진다 → 발산.
GPS 터널 문제 = 사실 이 문제.

이번 회차 핵심 다섯 개

1. **두 층**: 명목(비선형 전파) + 오차(선형 KF) 분리
2. **15D 오차 상태**: $[\delta\boldsymbol{\phi}, \delta\boldsymbol{p}, \delta\boldsymbol{v}, \delta\boldsymbol{b}_g, \delta\boldsymbol{b}_a]$ (OpenVINS 순서)
3. **Right perturbation**: $\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi})$ — 3D KF 가능
4. **Inject & Reset**: 오차를 명목에 합치고 0으로 초기화
5. **Joseph form**: $\boldsymbol{P}^{+} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})^{\top} + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^{\top}$

한 줄 요약
ESKF = KF를 SO(3)에서 쓰기 위한 좌표 변환 트릭.
어렵지 않다 — 단지 “오차를 별도로 추적한다”는 것.

과제 4 (다음 시간 전)

1. 슬라이드 8의 F 행렬을 손으로 유도. 세 번째 행($\delta \boldsymbol{v}$)이 $-\hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{a}}]_\times$ 인 이유를 $\boldsymbol{v}$의 시간 미분 방정식에서 시작해서 보여라.
2. 슬라이드 9의 OpenVINS 코드와 슬라이드 8의 해석적 F를 비교. 어떤 항이 서로 대응하는가? (`F.block(0,9,3,3)` 등)
3. (선택) Python으로 1D 위치+속도 문제를 ESKF 구조로 재구현. 명목 상태와 오차 상태를 분리해서, Inject & Reset 후 오차 분산이 0이 됨을 확인.
4. (선택) Solà 2017 (arXiv:1812.01537) §5 읽기 — 오차 상태 정의 부분.

다음 시간 예고 — 5회차: IMU 물리 + ESKF Predict

IMU 센서 모델, 바이어스, 노이즈, F/G 행렬 완전 유도

- 가속도계 · 자이로스코프 측정 모델 (Allan deviation, IMU grade)
- 연속/이산 F 행렬 유도 전 과정
- 노이즈 공분산 G 행렬과 $\boldsymbol{Q}_d$ 계산
- IMU Preintegration 개념 소개 (Forster 2017)
- OpenVINS `Propagator.cpp` 전체 흐름 따라가기

미리 생각해볼 것
IMU 100Hz, 카메라 10Hz일 때, 카메라 두 frame 사이에 IMU가 10개.
F를 10번 곱하는 것과 한 번에 유도하는 것은 같은가?

질문 환영

오차 상태를 0 근처에 묶어두는 것 — 이것이 ESKF

멀리 보기: 이 코스가 도달하는 한 줄

$$\text{ESKF} \subset \text{IEKF} \equiv \text{Gauss-Newton on MAP} \subset \text{Factor Graph}$$

오늘 본 ESKF는 이 부등식의 가장 왼쪽. 7회차에서 "$\equiv$" 등치성을 보일 것이고,
8회차(MSCKF)와 9회차(FAST-LIO)가 이 부등식의 오른쪽으로 자연스럽게 이어진다.
Bell & Cathey (IEEE TAC 1993).

5회차: IMU 물리 + ESKF Predict

지난 시간 한 줄

ESKF = 명목 상태(비선형 전파) + 오차 상태(선형 KF) 두 층.
회전 오차 $\delta\boldsymbol{\phi} \in \mathbb{R}^3$가 핵심.

오늘:

- IMU가 실제로 무엇을 측정하는가
- 측정값에서 바이어스와 노이즈를 어떻게 모델링하는가
- ESKF Predict의 F, G 행렬을 처음부터 유도
- OpenVINS 코드와 대조하여 확인

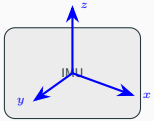
IMU — 6축 센서

가속도계 (Accelerometer):

- Body frame의 비중력 가속도 측정
- $\boldsymbol{a}_m = \boldsymbol{R}^\top(\boldsymbol{a}_W - \boldsymbol{g}_W) + \boldsymbol{b}_a + \boldsymbol{n}_a$
  ($\boldsymbol{a}_W, \boldsymbol{g}_W$: World frame 가속도/중력)
- 정적에서: 중력 $\boldsymbol{g}$ 반대 방향을 가리킴

자이로스코프 (Gyroscope):

- Body frame의 각속도 측정
- $\boldsymbol{\omega}_m = \boldsymbol{\omega}_{\text{true}} + \boldsymbol{b}_g + \boldsymbol{n}_g$
- 코리올리 효과(MEMS) 또는 광 간섭(광섬유, 링레이저)



Body frame
(B)

IMU 측정 모델 — 핵심 방정식

가속도계

$$\boldsymbol{a}_m = \underbrace{\boldsymbol{R}^\top(\boldsymbol{a}_W - \boldsymbol{g}_W)}_{\text{진짜 비중력 가속도 (Body frame)}} + \underbrace{\boldsymbol{b}_a}_{\text{바이어스}} + \underbrace{\boldsymbol{n}_a}_{\text{화이트 노이즈}}$$

($\boldsymbol{a}_W$: World frame 운동 가속도, $\boldsymbol{g}_W$: World frame 중력. ENU/NED 모두 가능, 부호만 일관)

자이로스코프

$$\boldsymbol{\omega}_m = \underbrace{\boldsymbol{\omega}}_{\text{진짜 각속도}} + \underbrace{\boldsymbol{b}_g}_{\text{바이어스}} + \underbrace{\boldsymbol{n}_g}_{\text{화이트 노이즈}}$$

IMU 측정 모델 — 핵심 방정식

가속도계

$$\boldsymbol{a}_m = \underbrace{\boldsymbol{R}^\top(\boldsymbol{a}_W - \boldsymbol{g}_W)}_{\text{진짜 비중력 가속도 (Body frame)}} + \underbrace{\boldsymbol{b}_a}_{\text{바이어스}} + \underbrace{\boldsymbol{n}_a}_{\text{화이트 노이즈}}$$

($\boldsymbol{a}_W$: World frame 운동 가속도, $\boldsymbol{g}_W$: World frame 중력. ENU/NED 모두 가능, 부호만 일관)

자이로스코프

$$\boldsymbol{\omega}_m = \underbrace{\boldsymbol{\omega}}_{\text{진짜 각속도}} + \underbrace{\boldsymbol{b}_g}_{\text{바이어스}} + \underbrace{\boldsymbol{n}_g}_{\text{화이트 노이즈}}$$

de-biased 측정값 (바이어스 추정치로 보정):

$$\tilde{\boldsymbol{a}} = \boldsymbol{a}_m - \hat{\boldsymbol{b}}_a, \qquad \tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\boldsymbol{b}}_g$$

바이어스 모델 — 임의 보행

바이어스는 느리게 변하는 확률 과정으로 모델링:

$$\dot{\boldsymbol{b}}_g = \boldsymbol{n}_{bg}, \quad \dot{\boldsymbol{b}}_a = \boldsymbol{n}_{ba}$$

$\boldsymbol{n}_{bg}, \boldsymbol{n}_{ba}$: 화이트 노이즈 (분산 $\sigma_{bg}^2, \sigma_{ba}^2$)

결과
바이어스는 **적분된 노이즈**
= Brownian motion
= ESKF 상태로 추정 가능

이산화 (Δt):

$$\boldsymbol{b}_{k+1} \approx \boldsymbol{b}_k + \boldsymbol{n}_{bg}\,\Delta t$$

분산: $\sigma_{\text{disc}}^2 = \sigma_{bg}^2\,\Delta t$

바이어스 모델 — 임의 보행

바이어스는 느리게 변하는 확률 과정으로 모델링:

$$\dot{\boldsymbol{b}}_g = \boldsymbol{n}_{bg}, \quad \dot{\boldsymbol{b}}_a = \boldsymbol{n}_{ba}$$

$\boldsymbol{n}_{bg}, \boldsymbol{n}_{ba}$: 화이트 노이즈 (분산 $\sigma_{bg}^2, \sigma_{ba}^2$)

이산화 (Δt):

$$\boldsymbol{b}_{k+1} \approx \boldsymbol{b}_k + \boldsymbol{n}_{bg}\,\Delta t$$

분산: $\sigma_{\text{disc}}^2 = \sigma_{bg}^2\,\Delta t$

결과
바이어스는 **적분된 노이즈**
= Brownian motion
= ESKF 상태로 추정 가능

왜 ESKF가 바이어스를 추정할 수 있는가
KF는 위치 측정 정보가 들어올 때, 공분산 교차항 $\boldsymbol{P}_{p,b}$를 통해 바이어스 오차를 역방향으로 업데이트.

노이즈 파라미터 — IMU Datasheet에서 읽기

파라미터	기호	단위
Gyro 각도 랜덤 보행 (ARW)	σ_g	deg/s/$\sqrt{\text{Hz}}$ 또는 rad/s/$\sqrt{\text{Hz}}$
Gyro 바이어스 불안정성	σ_{bg}	deg/hr
Accel 속도 랜덤 보행 (VRW)	σ_a	m/s^2/$\sqrt{\text{Hz}}$
Accel 바이어스 불안정성	σ_{ba}	μg 또는 m/s^2

노이즈 파라미터 — IMU Datasheet에서 읽기

파라미터	기호	단위
Gyro 각도 랜덤 보행 (ARW)	σ_g	deg/s/$\sqrt{\text{Hz}}$ 또는 rad/s/$\sqrt{\text{Hz}}$
Gyro 바이어스 불안정성	σ_{bg}	deg/hr
Accel 속도 랜덤 보행 (VRW)	σ_a	m/s^2/$\sqrt{\text{Hz}}$
Accel 바이어스 불안정성	σ_{ba}	μg 또는 m/s^2

Allan Deviation
로그-로그 그래프에서 기울기 $-1/2$ 구간 = 화이트 노이즈 (σ_g, σ_a)
기울기 0 구간 = 바이어스 불안정성 (σ_{bg}, σ_{ba})
기울기 $+1/2$ 구간 = 속도/각도 랜덤 보행 (더 장기 드리프트)

OpenVINS 파라미터 예: `gyroscope_noise_density: 1.6968e-04`, `gyroscope_random_walk: 1.9393e-05`

IMU 측정값으로 명목 상태를 전파:

$$\dot{\hat{\boldsymbol{p}}} = \hat{\boldsymbol{v}}$$

$$\dot{\hat{\boldsymbol{v}}} = \hat{\boldsymbol{R}}\,\tilde{\boldsymbol{a}} + \boldsymbol{g} \quad \text{(중력 World frame으로 더함)}$$

$$\dot{\hat{\boldsymbol{R}}} = \hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{\omega}}]_\times$$

$$\dot{\hat{\boldsymbol{b}}}_g = 0$$

$$\dot{\hat{\boldsymbol{b}}}_a = 0$$

명목 상태 이산화 (Euler)

연속시간 동역학을 Δt 간격으로 적분:

$$\hat{\boldsymbol{v}}_{k+1} = \hat{\boldsymbol{v}}_k + (\hat{\boldsymbol{R}}_k\,\tilde{\boldsymbol{a}}_k + \boldsymbol{g})\,\Delta t$$

$$\hat{\boldsymbol{p}}_{k+1} = \hat{\boldsymbol{p}}_k + \hat{\boldsymbol{v}}_k\,\Delta t + \tfrac{1}{2}(\hat{\boldsymbol{R}}_k\,\tilde{\boldsymbol{a}}_k + \boldsymbol{g})\Delta t^2$$

$$\hat{\boldsymbol{R}}_{k+1} = \hat{\boldsymbol{R}}_k\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k\,\Delta t)$$

명목 상태 이산화 (Euler)

연속시간 동역학을 Δt 간격으로 적분:

$$\hat{\boldsymbol{v}}_{k+1} = \hat{\boldsymbol{v}}_k + (\hat{\boldsymbol{R}}_k\,\tilde{\boldsymbol{a}}_k + \boldsymbol{g})\,\Delta t$$
$$\hat{\boldsymbol{p}}_{k+1} = \hat{\boldsymbol{p}}_k + \hat{\boldsymbol{v}}_k\,\Delta t + \tfrac{1}{2}(\hat{\boldsymbol{R}}_k\,\tilde{\boldsymbol{a}}_k + \boldsymbol{g})\Delta t^2$$
$$\hat{\boldsymbol{R}}_{k+1} = \hat{\boldsymbol{R}}_k\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k\,\Delta t)$$

실제 구현
실제 코드는 Runge-Kutta 4차 사용 (더 정확). OpenVINS: `Propagator.cpp`.

오차 상태 동역학 유도 — $\delta\boldsymbol{\phi}$

진짜 회전: $\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi})$. 엄밀히 $\frac{d}{dt}\mathrm{Exp}(\delta\boldsymbol{\phi}) = \mathrm{Exp}(\delta\boldsymbol{\phi})\,\left[J_r(\delta\boldsymbol{\phi})\,\delta\dot{\boldsymbol{\phi}}\right]_\times$.

사용하는 두 근사 (small $\delta\boldsymbol{\phi}$): (i) $\mathrm{Exp}(\delta\boldsymbol{\phi}) \approx \boldsymbol{I}$, (ii) $J_r(\delta\boldsymbol{\phi}) \approx \boldsymbol{I}$.

$$\dot{\boldsymbol{R}} = \hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{\omega}}]_\times\,\mathrm{Exp}(\delta\boldsymbol{\phi}) + \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi})\left[J_r\delta\dot{\boldsymbol{\phi}}\right]_\times \overset{(i),(ii)}{\approx} \hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{\omega}}]_\times + \hat{\boldsymbol{R}}\left[\delta\dot{\boldsymbol{\phi}}\right]_\times$$

진짜 회전 방정식:

$$\dot{\boldsymbol{R}} = \boldsymbol{R}\,[\boldsymbol{\omega}_{\text{true}}]_\times = \hat{\boldsymbol{R}}\,[\boldsymbol{\omega}_{\text{true}}]_\times = \hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{\omega}} - \delta\boldsymbol{b}_g - \boldsymbol{n}_g]_\times$$

오차 상태 동역학 유도 — $\delta\boldsymbol{\phi}$

진짜 회전: $\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi})$. 엄밀히 $\frac{d}{dt}\mathrm{Exp}(\delta\boldsymbol{\phi}) = \mathrm{Exp}(\delta\boldsymbol{\phi})\,\left[J_r(\delta\boldsymbol{\phi})\,\delta\dot{\boldsymbol{\phi}}\right]_\times$.

사용하는 두 근사 (small $\delta\boldsymbol{\phi}$): (i) $\mathrm{Exp}(\delta\boldsymbol{\phi}) \approx \boldsymbol{I}$, (ii) $J_r(\delta\boldsymbol{\phi}) \approx \boldsymbol{I}$.

$$\dot{\boldsymbol{R}} = \hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{\omega}}]_\times\;\mathrm{Exp}(\delta\boldsymbol{\phi}) + \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi})\left[J_r\delta\dot{\boldsymbol{\phi}}\right]_\times \overset{(i),(ii)}{\approx} \hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{\omega}}]_\times + \hat{\boldsymbol{R}}\left[\delta\dot{\boldsymbol{\phi}}\right]_\times$$

진짜 회전 방정식:

$$\dot{\boldsymbol{R}} = \boldsymbol{R}\,[\boldsymbol{\omega}_{\text{true}}]_\times = \hat{\boldsymbol{R}}\,[\boldsymbol{\omega}_{\text{true}}]_\times = \hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{\omega}} - \delta\boldsymbol{b}_g - \boldsymbol{n}_g]_\times$$

같다고 놓으면:

$$\hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{\omega}}]_\times + \hat{\boldsymbol{R}}\left[\delta\dot{\boldsymbol{\phi}}\right]_\times = \hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{\omega}}]_\times - \hat{\boldsymbol{R}}\,[\delta\boldsymbol{b}_g]_\times - \hat{\boldsymbol{R}}\,[\boldsymbol{n}_g]_\times$$

$$\boxed{\delta\dot{\boldsymbol{\phi}} = -\,[\tilde{\boldsymbol{\omega}}]_\times\,\delta\boldsymbol{\phi} - \delta\boldsymbol{b}_g - \boldsymbol{n}_g}$$

오차 상태 동역학 유도 — $\delta \boldsymbol{v}$

진짜 속도 미분:

$$\dot{\boldsymbol{v}} = \boldsymbol{R}\boldsymbol{a}_{\text{true}} + \boldsymbol{g} = \boldsymbol{R}(\tilde{\boldsymbol{a}} - \delta\boldsymbol{b}_a - \boldsymbol{n}_a) + \boldsymbol{g}$$

명목 속도 미분:

$$\dot{\hat{\boldsymbol{v}}} = \hat{\boldsymbol{R}}\,\tilde{\boldsymbol{a}} + \boldsymbol{g}$$

오차 상태 동역학 유도 — $\delta \boldsymbol{v}$

진짜 속도 미분:

$$\dot{\boldsymbol{v}} = \boldsymbol{R}\boldsymbol{a}_{\text{true}} + \boldsymbol{g} = \boldsymbol{R}(\tilde{\boldsymbol{a}} - \delta\boldsymbol{b}_a - \boldsymbol{n}_a) + \boldsymbol{g}$$

명목 속도 미분:

$$\dot{\hat{\boldsymbol{v}}} = \hat{\boldsymbol{R}}\,\tilde{\boldsymbol{a}} + \boldsymbol{g}$$

차이:

$$\begin{aligned}
\dot{\delta\boldsymbol{v}} &= \dot{\boldsymbol{v}} - \dot{\hat{\boldsymbol{v}}} = \boldsymbol{R}(\tilde{\boldsymbol{a}} - \delta\boldsymbol{b}_a - \boldsymbol{n}_a) - \hat{\boldsymbol{R}}\,\tilde{\boldsymbol{a}} \\
&\approx (\hat{\boldsymbol{R}} + \hat{\boldsymbol{R}}\,[\delta\boldsymbol{\phi}]_\times)(\tilde{\boldsymbol{a}} - \delta\boldsymbol{b}_a - \boldsymbol{n}_a) - \hat{\boldsymbol{R}}\,\tilde{\boldsymbol{a}} \\
&= \hat{\boldsymbol{R}}\,[\delta\boldsymbol{\phi}]_\times\,\tilde{\boldsymbol{a}} - \hat{\boldsymbol{R}}\,\delta\boldsymbol{b}_a - \hat{\boldsymbol{R}}\,\boldsymbol{n}_a \\
&= -\hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{a}}]_\times\,\delta\boldsymbol{\phi} - \hat{\boldsymbol{R}}\,\delta\boldsymbol{b}_a - \hat{\boldsymbol{R}}\,\boldsymbol{n}_a
\end{aligned}$$

$$\boxed{\dot{\delta\boldsymbol{v}} = -\hat{\boldsymbol{R}}\,[\tilde{\boldsymbol{a}}]_\times\,\delta\boldsymbol{\phi} - \hat{\boldsymbol{R}}\,\delta\boldsymbol{b}_a - \hat{\boldsymbol{R}}\,\boldsymbol{n}_a}$$

연속시간 F, G 행렬 (15×15, 15×12)

상태 순서: $\delta \boldsymbol{x} = [\delta \boldsymbol{\phi},\, \delta \boldsymbol{p},\, \delta \boldsymbol{v},\, \delta \boldsymbol{b}_g,\, \delta \boldsymbol{b}_a]^\top$

$$\boldsymbol{F} = \begin{pmatrix} -[\tilde{\boldsymbol{\omega}}]_\times & 0 & 0 & -\boldsymbol{I} & 0 \\ 0 & 0 & \boldsymbol{I} & 0 & 0 \\ -\hat{\boldsymbol{R}}[\tilde{\boldsymbol{a}}]_\times & 0 & 0 & 0 & -\hat{\boldsymbol{R}} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

노이즈 순서: $\boldsymbol{n} = [\boldsymbol{n}_g, \boldsymbol{n}_a, \boldsymbol{n}_{bg}, \boldsymbol{n}_{ba}]^\top \in \mathbb{R}^{12}$

$$\boldsymbol{G} = \begin{pmatrix} -\boldsymbol{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -\hat{\boldsymbol{R}} & 0 & 0 \\ 0 & 0 & \boldsymbol{I} & 0 \\ 0 & 0 & 0 & \boldsymbol{I} \end{pmatrix}$$

이산화 — Euler vs 행렬 지수

1차 Euler 이산화:

$$\boldsymbol{F}_d \approx \boldsymbol{I} + \boldsymbol{F}\,\Delta t$$

$$\boldsymbol{Q}_d \approx \boldsymbol{G}\,\boldsymbol{Q}_c\,\boldsymbol{G}^\top\,\Delta t$$

이산화 — Euler vs 행렬 지수

1차 Euler 이산화:

$$\boldsymbol{F}_d \approx \boldsymbol{I} + \boldsymbol{F}\,\Delta t$$

$$\boldsymbol{Q}_d \approx \boldsymbol{G}\,\boldsymbol{Q}_c\,\boldsymbol{G}^\top\,\Delta t$$

2차 Cayley-Hamilton (Van Loan 방법):

$$\boldsymbol{F}_d \approx \boldsymbol{I} + \boldsymbol{F}\,\Delta t + \tfrac{1}{2}\boldsymbol{F}^2\,\Delta t^2$$

$$\boldsymbol{Q}_d = \int_0^{\Delta t} e^{\boldsymbol{F}\tau}\boldsymbol{G}\boldsymbol{Q}_c\boldsymbol{G}^\top e^{\boldsymbol{F}^\top\tau}\,d\tau$$

이산화 — Euler vs 행렬 지수

1차 Euler 이산화:

$$\boldsymbol{F}_d \approx \boldsymbol{I} + \boldsymbol{F}\,\Delta t$$

$$\boldsymbol{Q}_d \approx \boldsymbol{G}\,\boldsymbol{Q}_c\,\boldsymbol{G}^\top\,\Delta t$$

2차 Cayley-Hamilton (Van Loan 방법):

$$\boldsymbol{F}_d \approx \boldsymbol{I} + \boldsymbol{F}\,\Delta t + \tfrac{1}{2}\boldsymbol{F}^2\,\Delta t^2$$

$$\boldsymbol{Q}_d = \int_0^{\Delta t} e^{\boldsymbol{F}\tau}\boldsymbol{G}\boldsymbol{Q}_c\boldsymbol{G}^\top e^{\boldsymbol{F}^\top\tau}\,d\tau$$

실용적 선택
IMU 100Hz, $\Delta t = 0.01$s일 때 1차 Euler로 충분한 경우 많음.
OpenVINS는 고차 근사 사용 (Xi_1..Xi_4).

OpenVINS Propagator — 이산 Qd 계산

ov_msckf/src/update/Propagator.cpp

```
// Continuous noise covariance (12x12): [n_g, n_ba, n_bg, n_a] order
Eigen::MatrixXd Qc = Eigen::MatrixXd::Zero(12,12);
Qc.block(0,0,3,3) = pow(imu_data.gyro_noise,2)*Eigen::Matrix3d::Identity();
Qc.block(3,3,3,3) = pow(imu_data.accel_noise,2)*Eigen::Matrix3d::Identity();
Qc.block(6,6,3,3) = pow(imu_data.gyro_walk,2)*Eigen::Matrix3d::Identity();
Qc.block(9,9,3,3) = pow(imu_data.accel_walk,2)*Eigen::Matrix3d::Identity();

// Discrete Qd using Van Loan / higher-order Taylor
// Xi_1 = I*dt + 0.5*F_33*dt^2 + ...  (computed analytically)
Qd = G * Qc * G.transpose() * dt;  // simplified; code uses Xi matrices
```

출처: github.com/rpng/open_vins — Propagator.cpp, GPLv3 라이선스

연속 잡음 밀도 σ_g^2 → 이산 분산 $\sigma_g^2/\Delta t$ (샘플링 스펙 확인 필요).

ESKF Predict — 전체 알고리즘

IMU 측정 $(\boldsymbol{\omega}_m, \boldsymbol{a}_m)$이 도착할 때마다:

1. $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\boldsymbol{b}}_g,\ \tilde{\boldsymbol{a}} = \boldsymbol{a}_m - \hat{\boldsymbol{b}}_a$
2. **명목 상태 적분** (Euler 또는 RK4):

$$\hat{\boldsymbol{v}} \leftarrow \hat{\boldsymbol{v}} + (\hat{\boldsymbol{R}}\tilde{\boldsymbol{a}} + \boldsymbol{g})\,\Delta t$$
$$\hat{\boldsymbol{p}} \leftarrow \hat{\boldsymbol{p}} + \hat{\boldsymbol{v}}\,\Delta t$$
$$\hat{\boldsymbol{R}} \leftarrow \hat{\boldsymbol{R}}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}\,\Delta t)$$

3. **F, G 계산** ($\hat{\boldsymbol{R}}, \tilde{\boldsymbol{a}}, \tilde{\boldsymbol{\omega}}$ 사용)
4. **이산** $\boldsymbol{F}_d, \boldsymbol{Q}_d$ **계산**
5. **공분산 전파**: $\boldsymbol{P} \leftarrow \boldsymbol{F}_d \boldsymbol{P} \boldsymbol{F}_d^\top + \boldsymbol{Q}_d$

ESKF Predict — 전체 알고리즘

IMU 측정 $(\boldsymbol{\omega}_m, \boldsymbol{a}_m)$이 도착할 때마다:

1. $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\boldsymbol{b}}_g,\ \tilde{\boldsymbol{a}} = \boldsymbol{a}_m - \hat{\boldsymbol{b}}_a$
2. **명목 상태 적분** (Euler 또는 RK4):

$$\hat{\boldsymbol{v}} \leftarrow \hat{\boldsymbol{v}} + (\hat{\boldsymbol{R}}\,\tilde{\boldsymbol{a}} + \boldsymbol{g})\,\Delta t$$
$$\hat{\boldsymbol{p}} \leftarrow \hat{\boldsymbol{p}} + \hat{\boldsymbol{v}}\,\Delta t$$
$$\hat{\boldsymbol{R}} \leftarrow \hat{\boldsymbol{R}}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}\,\Delta t)$$

3. **F, G 계산** ($\hat{\boldsymbol{R}}, \tilde{\boldsymbol{a}}, \tilde{\boldsymbol{\omega}}$ 사용)
4. **이산 $\boldsymbol{F}_d, \boldsymbol{Q}_d$ 계산**
5. **공분산 전파**: $\boldsymbol{P} \leftarrow \boldsymbol{F}_d \boldsymbol{P} \boldsymbol{F}_d^\top + \boldsymbol{Q}_d$

주의
명목 상태 업데이트(2번)는 비선형이고, 공분산 전파(5번)는 선형이다.
이 분리가 ESKF의 핵심이다.

IMU 등급 — MEMS vs 전술급

등급	ARW (deg/$\sqrt{\text{hr}}$)	바이어스 안정성 (deg/hr)	대표 예시
소비자 MEMS	> 0.5	> 100	MPU-9250, ICM-42688
산업용 MEMS	0.1–0.5	10–100	BMI088, ADIS16470
전술급	0.01–0.1	1–10	Analog Devices ADIS16505
항법급 (RLG/FOG)	< 0.01	< 0.1	HG1700, iXblue

IMU 등급 — MEMS vs 전술급

등급	ARW (deg/$\sqrt{\text{hr}}$)	바이어스 안정성 (deg/hr)	대표 예시
소비자 MEMS	> 0.5	> 100	MPU-9250, ICM-42688
산업용 MEMS	0.1–0.5	10–100	BMI088, ADIS16470
전술급	0.01–0.1	1–10	Analog Devices ADIS16505
항법급 (RLG/FOG)	< 0.01	< 0.1	HG1700, iXblue

의미

MEMS IMU + GPS 없이 순수 적분(Dead reckoning): 1분 후 위치 오차 $\sim$ 100m (소비자급) vs. $\sim$ 1m (전술급).
그래서 GNSS/카메라/LiDAR 보정이 필수.

IMU Preintegration 개념 (Forster 2017)

문제: 카메라 두 keyframe 사이에 IMU가 100개 → F를 100번 곱해야 하는가?

해법: Preintegration — IMU 측정을 한 번에 "요약"

$$\Delta \boldsymbol{R}_{ij} = \prod_{k=i}^{j-1} \mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \, \Delta t)$$

$$\Delta \boldsymbol{v}_{ij} = \sum_{k=i}^{j-1} \Delta \boldsymbol{R}_{ik} \, \tilde{\boldsymbol{a}}_k \, \Delta t$$

$$\Delta \boldsymbol{p}_{ij} = \sum_{k=i}^{j-1} \left(\Delta \boldsymbol{v}_{ik} \, \Delta t + \tfrac{1}{2} \Delta \boldsymbol{R}_{ik} \, \tilde{\boldsymbol{a}}_k \, \Delta t^2 \right)$$

IMU Preintegration 개념 (Forster 2017)

문제: 카메라 두 keyframe 사이에 IMU가 100개 → F를 100번 곱해야 하는가?

해법: Preintegration — IMU 측정을 한 번에 "요약"

$$\Delta \boldsymbol{R}_{ij} = \prod_{k=i}^{j-1} \mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \, \Delta t)$$

$$\Delta \boldsymbol{v}_{ij} = \sum_{k=i}^{j-1} \Delta \boldsymbol{R}_{ik} \, \tilde{\boldsymbol{a}}_k \, \Delta t$$

$$\Delta \boldsymbol{p}_{ij} = \sum_{k=i}^{j-1} \left(\Delta \boldsymbol{v}_{ik} \, \Delta t + \tfrac{1}{2} \Delta \boldsymbol{R}_{ik} \, \tilde{\boldsymbol{a}}_k \, \Delta t^2 \right)$$

핵심 이점
$\Delta \boldsymbol{R}_{ij}, \Delta \boldsymbol{v}_{ij}, \Delta \boldsymbol{p}_{ij}$는 바이어스만 바뀌면 선형 보정으로 재계산 가능.
GTSAM factor graph에서 IMU factor로 직접 사용 (Forster, RSS 2015).

estimator/src/factor/integration_base.h

```
void midPointIntegration(double dt,
    const Vector3d &acc_0, const Vector3d &gyr_0,
    const Vector3d &acc_1, const Vector3d &gyr_1,
    const Vector3d &delta_p,    // preintegrated p
    const Quaterniond &delta_q, // preintegrated q
    const Vector3d &delta_v,    // preintegrated v
    const Vector3d &linearized_ba,
    const Vector3d &linearized_bg,
    Vector3d &result_delta_p, ...)
{
    Vector3d w_x = 0.5 * (gyr_0 + gyr_1) - linearized_bg;  // de-biased
    Vector3d a_x = 0.5 * (acc_0 + acc_1) - linearized_ba;  // de-biased
    // ... F, V matrix build + covariance propagation
}
```

출처: github.com/HKUST-Aerial-Robotics/VINS-Mono — GPL-3.0

mid-point 적분 = IMU 두 sample의 평균을 한 step에 적분 (RK2 수준 정확도).

중력 벡터 — World frame과 Body frame

- World frame (NED): $\boldsymbol{g} = (0, 0, g)^\top$, $g \approx 9.81$ m/s²
- World frame (ENU): $\boldsymbol{g} = (0, 0, -g)^\top$
- 가속도계는 Body frame에서 측정: $\boldsymbol{a}_m = \hat{\boldsymbol{R}}^\top (\boldsymbol{a}_{\text{World}} - \boldsymbol{g}) + \boldsymbol{b}_a + \boldsymbol{n}_a$

중력 벡터 — World frame과 Body frame

- World frame (NED): $\boldsymbol{g} = (0, 0, g)^\top$, $g \approx 9.81$ m/s²
- World frame (ENU): $\boldsymbol{g} = (0, 0, -g)^\top$
- 가속도계는 Body frame에서 측정: $\boldsymbol{a}_m = \hat{\boldsymbol{R}}^\top (\boldsymbol{a}_{\text{World}} - \boldsymbol{g}) + \boldsymbol{b}_a + \boldsymbol{n}_a$

흔한 실수

명목 전파식 $\dot{\hat{\boldsymbol{v}}} = \hat{\boldsymbol{R}}\tilde{\boldsymbol{a}} + \boldsymbol{g}$에서
$\boldsymbol{g}$의 부호와 좌표계가 World frame인지 확인하지 않으면 발산.
OpenVINS는 ENU: $\boldsymbol{g} = [0, 0, -9.81]$.

중력 벡터 — World frame과 Body frame

- World frame (NED): $\boldsymbol{g} = (0, 0, g)^\top$, $g \approx 9.81$ m/s²
- World frame (ENU): $\boldsymbol{g} = (0, 0, -g)^\top$
- 가속도계는 Body frame에서 측정: $\boldsymbol{a}_m = \hat{\boldsymbol{R}}^\top (\boldsymbol{a}_{\text{World}} - \boldsymbol{g}) + \boldsymbol{b}_a + \boldsymbol{n}_a$

흔한 실수
명목 전파식 $\dot{\hat{\boldsymbol{v}}} = \hat{\boldsymbol{R}}\tilde{\boldsymbol{a}} + \boldsymbol{g}$에서
$\boldsymbol{g}$의 부호와 좌표계가 World frame인지 확인하지 않으면 발산.
OpenVINS는 ENU: $\boldsymbol{g} = [0, 0, -9.81]$.

자기 보정 (self-calibration)
초기 정지 상태에서 $\boldsymbol{a}_m \approx -\hat{\boldsymbol{R}}^\top \boldsymbol{g} \rightarrow \hat{\boldsymbol{R}}_0$ 추정 가능.
“정렬 단계”(alignment phase)라고 부른다.

이번 회차 핵심 다섯 개

1. 측정 모델: $\boldsymbol{a}_m = \hat{\boldsymbol{R}}^\top(\boldsymbol{a} - \boldsymbol{g}) + \boldsymbol{b}_a + \boldsymbol{n}_a,\ \boldsymbol{\omega}_m = \boldsymbol{\omega} + \boldsymbol{b}_g + \boldsymbol{n}_g$
2. F 행렬 $\dot{\delta\boldsymbol{\phi}} = -[\tilde{\boldsymbol{\omega}}]_\times \delta\boldsymbol{\phi} - \delta\boldsymbol{b}_g, \dot{\delta\boldsymbol{v}} = -\hat{\boldsymbol{R}}[\tilde{\boldsymbol{a}}]_\times \delta\boldsymbol{\phi} - \hat{\boldsymbol{R}}\delta\boldsymbol{b}_a$ 유도
3. G 행렬: 연속 노이즈를 오차 상태로 이어주는 경로
4. 이산화: $\boldsymbol{F}_d \approx \boldsymbol{I} + \boldsymbol{F}\Delta t, \boldsymbol{Q}_d \approx \boldsymbol{G}\boldsymbol{Q}_c\boldsymbol{G}^\top\Delta t$
5. Preintegration: 카메라 keyframe 사이 IMU를 한 번에 요약

한 줄 요약
IMU는 잡음+바이어스가 섞인 센서. ESKF는 그 오차를 선형 KF로 추적하며 보정한다.

과제 5 (다음 시간 전)

1. 슬라이드 8의 $\dot{\delta\boldsymbol{\phi}}$ 유도를 종이에 재현. 특히 $[\delta\boldsymbol{\phi}]_\times\ [\tilde{\boldsymbol{\omega}}]_\times \approx 0$ 근사가 어디서 쓰였는지 표시.
2. 슬라이드 9의 $\dot{\delta\boldsymbol{v}}$ 유도에서 $\boldsymbol{R} \approx \hat{\boldsymbol{R}}(\boldsymbol{I} + [\delta\boldsymbol{\phi}]_\times)$ 근사의 오차 크기를 $\|\delta\boldsymbol{\phi}\|^2$ 관점에서 추정.
3. (선택) MPU-9250 또는 BMI088 datasheet에서 gyro ARW와 bias instability를 찾아 ESKF $\boldsymbol{Q}_c$ 파라미터로 변환.
4. (선택) OpenVINS `Propagator.cpp`의 `compute_F_and_G_analytic()`를 읽고, 슬라이드 10의 F 행렬 각 블록과 대응 관계를 코멘트로 표시.

GPS, 휠 인코더, 카메라 측정을 ESKF에 어떻게 넣는가

- 일반 Update 공식: $\boldsymbol{z} = h(\hat{\boldsymbol{x}}) + \boldsymbol{H}\,\delta\boldsymbol{x} + \boldsymbol{v}$
- GPS: $\boldsymbol{H}_{\text{GPS}} = [0,\ \boldsymbol{I},\ 0,\ 0,\ 0]$ (위치 측정)
- 휠 속도: $\boldsymbol{H}_{\text{wheel}}$ 유도 (body frame velocity)
- 카메라 피처 잔차: OpenVINS MSCKF H matrix
- Mahalanobis 기반 이상값 제거 (outlier rejection)

미리 생각해볼 것
GPS는 World frame 위치를 준다. Body frame 속도만 주는 휠 인코더와 어떻게 합치는가?

질문 환영

IMU를 이해하면 ESKF Predict의 절반이 끝난다

Part II
Preintegration Jacobians

Forster 2017 TRO 완전 유도

바이어스 Jacobian · 공분산 전파 · Factor Residual

Preintegration 재방문 — 왜 Jacobian이 필요한가

기존 문제: 바이어스 추정이 업데이트되면 Δ를 처음부터 재적분해야 함.

Preintegration 재방문 — 왜 Jacobian이 필요한가

기존 문제: 바이어스 추정이 업데이트되면 Δ를 처음부터 재적분해야 함.

비용
키프레임 사이 수백 개의 IMU 샘플
$\Rightarrow$ 매 업데이트마다 $O(N)$ 재적분
$\Rightarrow$ 실시간 불가

해법 (Forster 2017)
바이어스 변화 δb에 대한
1차 Taylor 근사로 선형 보정
$\Rightarrow$ 재적분 없음

Preintegration 재방문 — 왜 Jacobian이 필요한가

기존 문제: 바이어스 추정이 업데이트되면 Δ를 처음부터 재적분해야 함.

비용
키프레임 사이 수백 개의 IMU 샘플
$\Rightarrow$ 매 업데이트마다 $O(N)$ 재적분
$\Rightarrow$ 실시간 불가

해법 (Forster 2017)
바이어스 변화 δb에 대한
1차 Taylor 근사로 선형 보정
$\Rightarrow$ 재적분 없음

핵심 근사식

$$\Delta R_{ij}(b_g + \delta b_g) \approx \Delta R_{ij}(b_g)\ \mathrm{Exp}\big(J_{b_g}^{\Delta R}\,\delta b_g\big)$$

$$\Delta v_{ij}(b + \delta b) \approx \Delta v_{ij}(b) + J_{b_g}^{\Delta v}\,\delta b_g + J_{b_a}^{\Delta v}\,\delta b_a$$

$$\Delta p_{ij}(b + \delta b) \approx \Delta p_{ij}(b) + J_{b_g}^{\Delta p}\,\delta b_g + J_{b_a}^{\Delta p}\,\delta b_a$$

Preintegration 재방문 — 왜 Jacobian이 필요한가

기존 문제: 바이어스 추정이 업데이트되면 Δ를 처음부터 재적분해야 함.

비용
키프레임 사이 수백 개의 IMU 샘플
$\Rightarrow$ 매 업데이트마다 $O(N)$ 재적분
$\Rightarrow$ 실시간 불가

해법 (Forster 2017)
바이어스 변화 δb에 대한
1차 Taylor 근사로 선형 보정
$\Rightarrow$ 재적분 없음

핵심 근사식

$$\Delta R_{ij}(b_g + \delta b_g) \approx \Delta R_{ij}(b_g)\ \mathrm{Exp}\big(J_{b_g}^{\Delta R}\,\delta b_g\big)$$

$$\Delta v_{ij}(b + \delta b) \approx \Delta v_{ij}(b) + J_{b_g}^{\Delta v}\,\delta b_g + J_{b_a}^{\Delta v}\,\delta b_a$$

$$\Delta p_{ij}(b + \delta b) \approx \Delta p_{ij}(b) + J_{b_g}^{\Delta p}\,\delta b_g + J_{b_a}^{\Delta p}\,\delta b_a$$

목표
이 **6개의 Jacobian 행렬**을 preintegration 동안 재귀적으로 같이 적분한다.

이산 시간 Preintegration 복습

두 키프레임 i, j 사이에 $N = j - i$개의 IMU 샘플이 있다고 하자.

Preintegrated Measurements

$$\Delta R_{ij} = \prod_{k=i}^{j-1} \mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \, \Delta t) \ \in \ \mathrm{SO}(3)$$

$$\Delta v_{ij} = \sum_{k=i}^{j-1} \Delta R_{ik} \ \tilde{\boldsymbol{a}}_k \, \Delta t \ \in \ \mathbb{R}^3$$

$$\Delta p_{ij} = \sum_{k=i}^{j-1} \left(\Delta v_{ik} \, \Delta t + \tfrac{1}{2} \, \Delta R_{ik} \ \tilde{\boldsymbol{a}}_k \, \Delta t^2 \right) \ \in \ \mathbb{R}^3$$

이산 시간 Preintegration 복습

두 키프레임 i, j 사이에 $N = j - i$개의 IMU 샘플이 있다고 하자.

Preintegrated Measurements

$$\Delta R_{ij} = \prod_{k=i}^{j-1} \mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \, \Delta t) \ \in \ \mathrm{SO}(3)$$

$$\Delta v_{ij} = \sum_{k=i}^{j-1} \Delta R_{ik} \, \tilde{\boldsymbol{a}}_k \, \Delta t \ \in \ \mathbb{R}^3$$

$$\Delta p_{ij} = \sum_{k=i}^{j-1} \left(\Delta v_{ik} \, \Delta t + \tfrac{1}{2} \, \Delta R_{ik} \, \tilde{\boldsymbol{a}}_k \, \Delta t^2 \right) \ \in \ \mathbb{R}^3$$

de-biased 측정값:

$$\tilde{\boldsymbol{\omega}}_k = \boldsymbol{\omega}_{m,k} - b_g, \qquad \tilde{\boldsymbol{a}}_k = \boldsymbol{a}_{m,k} - b_a$$

이산 시간 Preintegration 복습

두 키프레임 i, j 사이에 $N = j - i$개의 IMU 샘플이 있다고 하자.

Preintegrated Measurements

$$\Delta R_{ij} = \prod_{k=i}^{j-1} \mathrm{Exp}(\tilde{\omega}_k \, \Delta t) \ \in \ \mathrm{SO}(3)$$

$$\Delta v_{ij} = \sum_{k=i}^{j-1} \Delta R_{ik} \ \tilde{\boldsymbol{a}}_k \, \Delta t \ \in \ \mathbb{R}^3$$

$$\Delta p_{ij} = \sum_{k=i}^{j-1} \left(\Delta v_{ik} \, \Delta t + \tfrac{1}{2} \, \Delta R_{ik} \ \tilde{\boldsymbol{a}}_k \, \Delta t^2 \right) \ \in \ \mathbb{R}^3$$

de-biased 측정값:

$$\tilde{\boldsymbol{\omega}}_k = \boldsymbol{\omega}_{m,k} - b_g, \qquad \tilde{\boldsymbol{a}}_k = \boldsymbol{a}_{m,k} - b_a$$

재귀 관계 (한 스텝 전진)

$$\Delta R_{i,k+1} = \Delta R_{ik} \ \mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t), \quad \Delta v_{i,k+1} = \Delta v_{ik} + \Delta R_{ik} \ \tilde{\boldsymbol{a}}_k \, \Delta t$$

참고: $\Delta R_{ii} = I, \ \Delta v_{ii} = \mathbf{0}, \ \Delta p_{ii} = \mathbf{0}$

Noise 모델: IMU 연속 시간 $\rightarrow$ 이산 시간

이 강의의 부호 컨벤션 (이후 모든 슬라이드에 일관 적용)

$$\boldsymbol{\omega}_m = \boldsymbol{\omega} + b_g + \boldsymbol{n}_g, \qquad \boldsymbol{a}_m = \boldsymbol{R}^\top(\boldsymbol{a}_W - \boldsymbol{g}_W) + b_a + \boldsymbol{n}_a$$

노이즈가 **측정값에 더해지는** 형태. 따라서 "bias-제거된 추정치" $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - b_g$는 참값보다 $\boldsymbol{n}_g$만큼 큰 값: $\boldsymbol{\omega} = \tilde{\boldsymbol{\omega}} - \boldsymbol{n}_g$.
이 부호 규약 때문에 후속 유도(PI-21/22)에서 B_k에 **음의 부호**가 자연스럽게 등장한다.

Noise 모델: IMU 연속 시간 $\rightarrow$ 이산 시간

이 강의의 부호 컨벤션 (이후 모든 슬라이드에 일관 적용)

$$\boldsymbol{\omega}_m = \boldsymbol{\omega} + b_g + \boldsymbol{n}_g, \qquad \boldsymbol{a}_m = \boldsymbol{R}^\top(\boldsymbol{a}_W - \boldsymbol{g}_W) + b_a + \boldsymbol{n}_a$$

노이즈가 **측정값에 더해지는** 형태. 따라서 "bias-제거된 추정치" $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - b_g$는 참값보다 $\boldsymbol{n}_g$만큼 큰 값: $\boldsymbol{\omega} = \tilde{\boldsymbol{\omega}} - \boldsymbol{n}_g$.
이 부호 규약 때문에 후속 유도(PI-21/22)에서 B_k에 **음의 부호**가 자연스럽게 등장한다.

이산화 (Δt 적분 후)

$$\boldsymbol{\eta}_d^g \sim \mathcal{N}\left(0,\ \frac{\sigma_g^2}{\Delta t} I\right), \qquad \boldsymbol{\eta}_d^a \sim \mathcal{N}\left(0,\ \frac{\sigma_a^2}{\Delta t} I\right)$$

바이어스 임의 보행 이산화:

$$\boldsymbol{\eta}_{bg} \sim \mathcal{N}(0,\ \sigma_{bg}^2 \Delta t\, I), \qquad \boldsymbol{\eta}_{ba} \sim \mathcal{N}(0,\ \sigma_{ba}^2 \Delta t\, I)$$

Noise 모델: IMU 연속 시간 $\rightarrow$ 이산 시간

이 강의의 부호 컨벤션 (이후 모든 슬라이드에 일관 적용)

$$\boldsymbol{\omega}_m = \boldsymbol{\omega} + b_g + \boldsymbol{n}_g, \qquad \boldsymbol{a}_m = \boldsymbol{R}^\top(\boldsymbol{a}_W - \boldsymbol{g}_W) + b_a + \boldsymbol{n}_a$$

노이즈가 **측정값에 더해지는** 형태. 따라서 "bias-제거된 추정치" $\bar{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - b_g$는 참값보다 $\boldsymbol{n}_g$만큼 큰 값: $\boldsymbol{\omega} = \bar{\boldsymbol{\omega}} - \boldsymbol{n}_g$.
이 부호 규약 때문에 후속 유도(PI-21/22)에서 B_k에 **음의 부호**가 자연스럽게 등장한다.

이산화 (Δt 적분 후)

$$\boldsymbol{\eta}_d^g \sim \mathcal{N}\left(0,\ \frac{\sigma_g^2}{\Delta t} I\right), \qquad \boldsymbol{\eta}_d^a \sim \mathcal{N}\left(0,\ \frac{\sigma_a^2}{\Delta t} I\right)$$

바이어스 임의 보행 이산화:

$$\boldsymbol{\eta}_{bg} \sim \mathcal{N}(0,\ \sigma_{bg}^2 \Delta t\, I), \qquad \boldsymbol{\eta}_{ba} \sim \mathcal{N}(0,\ \sigma_{ba}^2 \Delta t\, I)$$

이산 노이즈 행렬 $Q_d \in \mathbb{R}^{12\times 12}$

$$Q_d = \mathrm{diag}\left(\frac{\sigma_g^2}{\Delta t} I_3,\ \frac{\sigma_a^2}{\Delta t} I_3,\ \sigma_{bg}^2 \Delta t\, I_3,\ \sigma_{ba}^2 \Delta t\, I_3\right)$$

순서: gyro noise | accel noise | gyro RW | accel RW

Noise 모델: IMU 연속 시간 $\rightarrow$ 이산 시간

이 강의의 부호 컨벤션 (이후 모든 슬라이드에 일관 적용)

$$\boldsymbol{\omega}_m = \boldsymbol{\omega} + b_g + \boldsymbol{n}_g, \qquad \boldsymbol{a}_m = \boldsymbol{R}^\top(\boldsymbol{a}_W - \boldsymbol{g}_W) + b_a + \boldsymbol{n}_a$$

노이즈가 **측정값에 더해지는** 형태. 따라서 "bias-제거된 추정치" $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - b_g$는 참값보다 $\boldsymbol{n}_g$만큼 큰 값: $\boldsymbol{\omega} = \tilde{\boldsymbol{\omega}} - \boldsymbol{n}_g$.
이 부호 규약 때문에 후속 유도(PI-21/22)에서 B_k에 **음의 부호**가 자연스럽게 등장한다.

이산화 (Δt 적분 후)

$$\boldsymbol{\eta}_d^g \sim \mathcal{N}\left(0,\ \frac{\sigma_g^2}{\Delta t} I\right), \qquad \boldsymbol{\eta}_d^a \sim \mathcal{N}\left(0,\ \frac{\sigma_a^2}{\Delta t} I\right)$$

바이어스 임의 보행 이산화:

$$\boldsymbol{\eta}_{bg} \sim \mathcal{N}(0,\ \sigma_{bg}^2 \Delta t\, I), \qquad \boldsymbol{\eta}_{ba} \sim \mathcal{N}(0,\ \sigma_{ba}^2 \Delta t\, I)$$

이산 노이즈 행렬 $Q_d \in \mathbb{R}^{12\times 12}$

$$Q_d = \mathrm{diag}\left(\frac{\sigma_g^2}{\Delta t} I_3,\ \frac{\sigma_a^2}{\Delta t} I_3,\ \sigma_{bg}^2 \Delta t\, I_3,\ \sigma_{ba}^2 \Delta t\, I_3\right)$$

순서: gyro noise | accel noise | gyro RW | accel RW

OpenVINS 파라미터 대응

`gyroscope_noise_density` $\rightarrow \sigma_g$, `accelerometer_noise_density` $\rightarrow \sigma_a$
`gyroscope_random_walk` $\rightarrow \sigma_{bg}$, `accelerometer_random_walk` $\rightarrow \sigma_{ba}$

Noise 모델: IMU 연속 시간 → 이산 시간

이 강의의 부호 컨벤션 (이후 모든 슬라이드에 일관 적용)

$$\boldsymbol{\omega}_m = \boldsymbol{\omega} + b_g + \boldsymbol{n}_g, \qquad \boldsymbol{a}_m = \boldsymbol{R}^\top(\boldsymbol{a}_W - \boldsymbol{g}_W) + b_a + \boldsymbol{n}_a$$

노이즈가 **측정값에 더해지는** 형태. 따라서 "bias-제거된 추정치" $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - b_g$는 참값보다 $\boldsymbol{n}_g$만큼 큰 값: $\boldsymbol{\omega} = \tilde{\boldsymbol{\omega}} - \boldsymbol{n}_g$.
이 부호 규약 때문에 후속 유도(PI-21/22)에서 B_k에 **음의 부호**가 자연스럽게 등장한다.

이산화 (Δt 적분 후)

$$\boldsymbol{\eta}_d^g \sim \mathcal{N}\left(0,\ \frac{\sigma_g^2}{\Delta t} I\right), \qquad \boldsymbol{\eta}_d^a \sim \mathcal{N}\left(0,\ \frac{\sigma_a^2}{\Delta t} I\right)$$

바이어스 임의 보행 이산화:

$$\boldsymbol{\eta}_{bg} \sim \mathcal{N}(0,\ \sigma_{bg}^2 \Delta t\, I), \qquad \boldsymbol{\eta}_{ba} \sim \mathcal{N}(0,\ \sigma_{ba}^2 \Delta t\, I)$$

이산 노이즈 행렬 $Q_d \in \mathbb{R}^{12\times 12}$

$$Q_d = \mathrm{diag}\left(\frac{\sigma_g^2}{\Delta t} I_3,\ \frac{\sigma_a^2}{\Delta t} I_3,\ \sigma_{bg}^2 \Delta t\, I_3,\ \sigma_{ba}^2 \Delta t\, I_3\right)$$

순서: gyro noise | accel noise | gyro RW | accel RW

OpenVINS 파라미터 대응

`gyroscope_noise_density` → σ_g, `accelerometer_noise_density` → σ_a
`gyroscope_random_walk` → σ_{bg}, `accelerometer_random_walk` → σ_{ba}

PSD 컨벤션 두 가지 — 데이터시트 단위에 매번 확인 필요

컨벤션	이산 분산
Continuous PSD (σ_g 단위 = rad/s/$\sqrt{\text{Hz}}$)	$\sigma_g^2/\Delta t$
Discrete (이미 Δt 적분, σ_g^d 단위 = rad/s)	$(\sigma_a^d)^2$
Allan variance (deg/$\sqrt{\text{h}}$ 등)	단위 변환 후 위 식 사용

이 강의는 Continuous PSD 컨벤션을 사용한다 (OpenVINS, Kalibr, EuRoC 데이터셋과 일치). Forster 2017 / VINS-Mono도 PSD 컨벤션이지만, 일부 코드는 이미 $\sigma_g \cdot \sqrt{\Delta t}$ 형태로 받기도 하니 IMU 캘리브레이션의 1순위 디버깅 포인트.

Preintegration 공분산 전파 — 재귀식

상태: 스텝 k 에서 (frame i에 상대적인)

$$\boldsymbol{\xi}_k = \left(\delta\boldsymbol{\phi}_k,\ \delta v_k,\ \delta p_k\right) \in \mathbb{R}^9$$

(회전 오차 $\delta\phi \in \mathbb{R}^3$, 속도 오차 $\delta v \in \mathbb{R}^3$, 위치 오차 $\delta p \in \mathbb{R}^3$)

Preintegration 공분산 전파 — 재귀식

상태: 스텝 k 에서 (frame i에 상대적인)

$$\boldsymbol{\xi}_k = \left(\delta\boldsymbol{\phi}_k,\ \delta v_k,\ \delta p_k\right) \in \mathbb{R}^9$$

(회전 오차 $\delta\phi \in \mathbb{R}^3$, 속도 오차 $\delta v \in \mathbb{R}^3$, 위치 오차 $\delta p \in \mathbb{R}^3$)

선형화 오차 동역학 (한 스텝)

$$\boldsymbol{\xi}_{k+1} = A_k\,\boldsymbol{\xi}_k + B_k\,\boldsymbol{\eta}_k$$

- $A_k \in \mathbb{R}^{9\times 9}$: 오차 상태 전이 행렬
- $B_k \in \mathbb{R}^{9\times 12}$: 노이즈 입력 행렬
- $\boldsymbol{\eta}_k \in \mathbb{R}^{12}$: $[\boldsymbol{\eta}_g^\top,\ \boldsymbol{\eta}_a^\top,\ \boldsymbol{\eta}_{bg}^\top,\ \boldsymbol{\eta}_{ba}^\top]^\top$

Preintegration 공분산 전파 — 재귀식

상태: 스텝 k 에서 (frame i에 상대적인)

$$\boldsymbol{\xi}_k = \left(\delta\boldsymbol{\phi}_k,\ \delta v_k,\ \delta p_k\right) \in \mathbb{R}^9$$

(회전 오차 $\delta\phi \in \mathbb{R}^3$, 속도 오차 $\delta v \in \mathbb{R}^3$, 위치 오차 $\delta p \in \mathbb{R}^3$)

선형화 오차 동역학 (한 스텝)

$$\boldsymbol{\xi}_{k+1} = A_k\,\boldsymbol{\xi}_k + B_k\,\boldsymbol{\eta}_k$$

- $A_k \in \mathbb{R}^{9\times 9}$: 오차 상태 전이 행렬
- $B_k \in \mathbb{R}^{9\times 12}$: 노이즈 입력 행렬
- $\boldsymbol{\eta}_k \in \mathbb{R}^{12}$: $[\boldsymbol{\eta}_g^\top,\ \boldsymbol{\eta}_a^\top,\ \boldsymbol{\eta}_{bg}^\top,\ \boldsymbol{\eta}_{ba}^\top]^\top$

공분산 재귀 전파

$$\boxed{\Sigma_{k+1} = A_k\,\Sigma_k\,A_k^\top + B_k\,Q_d\,B_k^\top}$$

초기 조건: $\Sigma_{ii} = 0_{9\times 9}$

Preintegration 공분산 전파 — 재귀식

상태: 스텝 k 에서 (frame i에 상대적인)

$$\boldsymbol{\xi}_k = \left(\delta\boldsymbol{\phi}_k,\ \delta v_k,\ \delta p_k\right) \in \mathbb{R}^9$$

(회전 오차 $\delta\phi \in \mathbb{R}^3$, 속도 오차 $\delta v \in \mathbb{R}^3$, 위치 오차 $\delta p \in \mathbb{R}^3$)

선형화 오차 동역학 (한 스텝)

$$\boldsymbol{\xi}_{k+1} = A_k\,\boldsymbol{\xi}_k + B_k\,\boldsymbol{\eta}_k$$

- $A_k \in \mathbb{R}^{9\times 9}$: 오차 상태 전이 행렬
- $B_k \in \mathbb{R}^{9\times 12}$: 노이즈 입력 행렬
- $\boldsymbol{\eta}_k \in \mathbb{R}^{12}$: $[\boldsymbol{\eta}_g^\top,\ \boldsymbol{\eta}_a^\top,\ \boldsymbol{\eta}_{bg}^\top,\ \boldsymbol{\eta}_{ba}^\top]^\top$

공분산 재귀 전파

$$\boxed{\Sigma_{k+1} = A_k\,\Sigma_k\,A_k^\top + B_k\,Q_d\,B_k^\top}$$

초기 조건: $\Sigma_{ii} = 0_{9\times 9}$

이 식을 $k = i$ 부터 $k = j-1$ 까지 반복하면 $\Sigma_{ij} \in \mathbb{R}^{9\times 9}$ 획득.

A_k 행렬 유도 — 회전 오차 부분

명목 회전 전파:

$$\Delta R_{i,k+1} = \Delta R_{ik}\ \mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t)$$

실제 회전 (노이즈 포함):

$$\Delta \tilde{R}_{i,k+1} = \Delta R_{ik}\ \mathrm{Exp}\big((\tilde{\boldsymbol{\omega}}_k - \boldsymbol{\eta}_g)\Delta t\big)$$

A_k 행렬 유도 — 회전 오차 부분

명목 회전 전파:

$$\Delta R_{i,k+1} = \Delta R_{ik}\ \mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t)$$

실제 회전 (노이즈 포함):

$$\Delta \tilde{R}_{i,k+1} = \Delta R_{ik}\ \mathrm{Exp}\big((\tilde{\boldsymbol{\omega}}_k - \boldsymbol{\eta}_g)\Delta t\big)$$

오차 정의: $\Delta R_{ik} = \Delta \tilde{R}_{ik}\ \mathrm{Exp}(-\delta\phi_k)$. BCH 근사를 적용하면:

회전 오차 전파

$$\delta\phi_{k+1} = \mathrm{Exp}\big(-\tilde{\omega}_k \Delta t\big)\,\delta\phi_k \ + \ J_r^{-1}(\tilde{\omega}_k \Delta t)\,\Delta t\,\boldsymbol{\eta}_g$$

A_k 행렬 유도 — 회전 오차 부분

명목 회전 전파:

$$\Delta R_{i,k+1} = \Delta R_{ik}\ \mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t)$$

실제 회전 (노이즈 포함):

$$\Delta \tilde{R}_{i,k+1} = \Delta R_{ik}\ \mathrm{Exp}\big((\tilde{\boldsymbol{\omega}}_k - \boldsymbol{\eta}_g)\Delta t\big)$$

오차 정의: $\Delta R_{ik} = \Delta \tilde{R}_{ik}\ \mathrm{Exp}(-\delta\phi_k)$. BCH 근사를 적용하면:

회전 오차 전파

$$\delta\phi_{k+1} = \mathrm{Exp}\big(-\tilde{\boldsymbol{\omega}}_k \Delta t\big)\,\delta\phi_k \;+\; {J_r}^{-1}(\tilde{\boldsymbol{\omega}}_k \Delta t)\,\Delta t\,\boldsymbol{\eta}_g$$

따라서 A_k의 회전 블록:

$$A_k[0{:}3,\ 0{:}3] = \mathrm{Exp}\big(-\tilde{\boldsymbol{\omega}}_k \Delta t\big)\ \approx\ I - [\tilde{\boldsymbol{\omega}}_k]_\times\, \Delta t$$

A_k 행렬 유도 — 회전 오차 부분

명목 회전 전파:

$$\Delta R_{i,k+1} = \Delta R_{ik}\ \mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t)$$

실제 회전 (노이즈 포함):

$$\Delta \tilde{R}_{i,k+1} = \Delta R_{ik}\ \mathrm{Exp}\big((\tilde{\boldsymbol{\omega}}_k - \boldsymbol{\eta}_g)\Delta t\big)$$

오차 정의: $\Delta R_{ik} = \Delta \tilde{R}_{ik}\ \mathrm{Exp}(-\delta\phi_k)$. BCH 근사를 적용하면:

회전 오차 전파

$$\delta\phi_{k+1} = \mathrm{Exp}\big(-\tilde{\omega}_k \Delta t\big)\,\delta\phi_k \ +\ J_r^{-1}(\tilde{\omega}_k \Delta t)\,\Delta t\,\boldsymbol{\eta}_g$$

따라서 A_k의 회전 블록:

$$A_k[0{:}3,\ 0{:}3] = \mathrm{Exp}\big(-\tilde{\boldsymbol{\omega}}_k \Delta t\big)\ \approx\ I - [\tilde{\boldsymbol{\omega}}_k]_\times\,\Delta t$$

J_r^{-1}가 여기서 처음으로 등장하는 이유

BCH 공식에서 두 회전의 합성 오차를 선형화할 때 Right Jacobian의 역행렬이 자연스럽게 나타남.

$J_r(\boldsymbol{\phi}) = I - \frac{1-\cos\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^2}\,[\boldsymbol{\phi}]_\times + \frac{\|\boldsymbol{\phi}\|-\sin\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^3}\,[\boldsymbol{\phi}]_\times^2$ (작은 각도: $J_r \approx I - \frac{1}{2}\,[\boldsymbol{\phi}]_\times$)

A_k 행렬 유도 — 속도/위치 부분 + 전체 구조

속도 오차 전파: $\Delta v_{i,k+1} = \Delta v_{ik} + \Delta R_{ik}\tilde{\boldsymbol{a}}_k \Delta t$ 선형화

$$\delta v_{k+1} = \delta v_k - \Delta R_{ik} \left[\tilde{\boldsymbol{a}}_k\right]_\times \delta\phi_k \, \Delta t$$

위치 오차 전파:

$$\delta p_{k+1} = \delta p_k + \delta v_k \, \Delta t - \tfrac{1}{2}\Delta R_{ik} \left[\tilde{\boldsymbol{a}}_k\right]_\times \delta\phi_k \, \Delta t^2$$

A_k 행렬 유도 — 속도/위치 부분 + 전체 구조

속도 오차 전파: $\Delta v_{i,k+1} = \Delta v_{ik} + \Delta R_{ik}\tilde{\boldsymbol{a}}_k \Delta t$ 선형화

$$\delta v_{k+1} = \delta v_k - \Delta R_{ik} \left[\tilde{\boldsymbol{a}}_k\right]_\times \delta\phi_k \, \Delta t$$

위치 오차 전파:

$$\delta p_{k+1} = \delta p_k + \delta v_k \, \Delta t - \tfrac{1}{2}\Delta R_{ik} \left[\tilde{\boldsymbol{a}}_k\right]_\times \delta\phi_k \, \Delta t^2$$

$A_k \in \mathbb{R}^{9\times 9}$ **전체 블록 구조**

$$A_k = \begin{pmatrix} \mathrm{Exp}(-\tilde{\boldsymbol{\omega}}_k \Delta t) & \mathbf{0} & \mathbf{0} \\ -\Delta R_{ik} \left[\tilde{\boldsymbol{a}}_k\right]_\times \Delta t & I & \mathbf{0} \\ -\tfrac{1}{2}\Delta R_{ik} \left[\tilde{\boldsymbol{a}}_k\right]_\times \Delta t^2 & I\,\Delta t & I \end{pmatrix}$$

행/열 순서: $(\delta\phi,\ \delta v,\ \delta p)$

A_k 행렬 유도 — 속도/위치 부분 + 전체 구조

속도 오차 전파: $\Delta v_{i,k+1} = \Delta v_{ik} + \Delta R_{ik}\tilde{\boldsymbol{a}}_k \Delta t$ 선형화

$$\delta v_{k+1} = \delta v_k - \Delta R_{ik} \left[\tilde{\boldsymbol{a}}_k\right]_\times \delta\phi_k \, \Delta t$$

위치 오차 전파:

$$\delta p_{k+1} = \delta p_k + \delta v_k \, \Delta t - \tfrac{1}{2}\Delta R_{ik} \left[\tilde{\boldsymbol{a}}_k\right]_\times \delta\phi_k \, \Delta t^2$$

$A_k \in \mathbb{R}^{9\times 9}$ **전체 블록 구조**

$$A_k = \begin{pmatrix} \mathrm{Exp}(-\tilde{\boldsymbol{\omega}}_k \Delta t) & \mathbf{0} & \mathbf{0} \\ -\Delta R_{ik} \left[\tilde{\boldsymbol{a}}_k\right]_\times \Delta t & I & \mathbf{0} \\ -\tfrac{1}{2}\Delta R_{ik} \left[\tilde{\boldsymbol{a}}_k\right]_\times \Delta t^2 & I\,\Delta t & I \end{pmatrix}$$

행/열 순서: $(\delta\phi,\ \delta v,\ \delta p)$

직관

- 대각: 각 상태가 자기 자신에게 직접 전달
- 아래 삼각: 회전 오차 $\rightarrow$ 속도/위치 오차 (cross-coupling)
- 선형 구조 덕분에 KF Riccati 전파 가능

B_k 행렬 — 노이즈 입력 행렬

$B_k \in \mathbb{R}^{9\times 12}$: 노이즈 벡터 $\boldsymbol{\eta} = [\boldsymbol{\eta}_g^\top, \boldsymbol{\eta}_a^\top, \boldsymbol{\eta}_{bg}^\top, \boldsymbol{\eta}_{ba}^\top]^\top$를 오차 상태로 매핑.

B_k 행렬 — 노이즈 입력 행렬

$B_k \in \mathbb{R}^{9\times 12}$: 노이즈 벡터 $\boldsymbol{\eta} = [\boldsymbol{\eta}_g^\top, \boldsymbol{\eta}_a^\top, \boldsymbol{\eta}_{bg}^\top, \boldsymbol{\eta}_{ba}^\top]^\top$를 오차 상태로 매핑.

블록 구조 (PI-3 부호 규약과 일관 — PI-21/22에서 엄밀 유도됨)

$$B_k = \begin{pmatrix} -J_r^{-1}(\tilde{\boldsymbol{\omega}}_k \Delta t)\,\Delta t & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\Delta R_{ik}\,\Delta t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\tfrac{1}{2}\Delta R_{ik}\,\Delta t^2 & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

열 순서: $(\boldsymbol{\eta}_g \mid \boldsymbol{\eta}_a \mid \boldsymbol{\eta}_{bg} \mid \boldsymbol{\eta}_{ba})$, 각 3열. 공분산 전파 $B_k Q_d B_k^\top$에서는 부호가 $\pm$ 두 번 곱해져 사라지므로 결과 covariance는 동일.

B_k 행렬 — 노이즈 입력 행렬

$B_k \in \mathbb{R}^{9\times 12}$: 노이즈 벡터 $\boldsymbol{\eta} = [\boldsymbol{\eta}_g^\top, \boldsymbol{\eta}_a^\top, \boldsymbol{\eta}_{bg}^\top, \boldsymbol{\eta}_{ba}^\top]^\top$를 오차 상태로 매핑.

블록 구조 (PI-3 부호 규약과 일관 — PI-21/22에서 엄밀 유도됨)

$$B_k = \begin{pmatrix} -J_r^{-1}(\tilde{\boldsymbol{\omega}}_k \Delta t)\,\Delta t & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\Delta R_{ik}\,\Delta t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\tfrac{1}{2}\Delta R_{ik}\,\Delta t^2 & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

열 순서: $(\boldsymbol{\eta}_g \mid \boldsymbol{\eta}_a \mid \boldsymbol{\eta}_{bg} \mid \boldsymbol{\eta}_{ba})$, 각 3열. 공분산 전파 $B_k Q_d B_k^\top$에서는 부호가 $\pm$ 두 번 곱해져 사라지므로 결과 covariance는 동일.

비제로 블록 설명:

- $B_k[0{:}3, 0{:}3] = -J_r^{-1}\Delta t$: gyro 노이즈 $\rightarrow$ 회전 오차
- $B_k[3{:}6, 3{:}6] = -\Delta R_{ik}\Delta t$: accel 노이즈 $\rightarrow$ 속도 오차
- $B_k[6{:}9, 3{:}6] = -\frac{1}{2}\Delta R_{ik}\Delta t^2$: accel 노이즈 $\rightarrow$ 위치 오차

B_k 행렬 — 노이즈 입력 행렬

$B_k \in \mathbb{R}^{9\times 12}$: 노이즈 벡터 $\boldsymbol{\eta} = [\boldsymbol{\eta}_g^\top, \boldsymbol{\eta}_a^\top, \boldsymbol{\eta}_{bg}^\top, \boldsymbol{\eta}_{ba}^\top]^\top$를 오차 상태로 매핑.

블록 구조 (PI-3 부호 규약과 일관 — PI-21/22에서 엄밀 유도됨)

$$B_k = \begin{pmatrix} -J_r^{-1}(\tilde{\boldsymbol{\omega}}_k \Delta t)\,\Delta t & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\Delta R_{ik}\,\Delta t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\tfrac{1}{2}\Delta R_{ik}\,\Delta t^2 & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

열 순서: $(\boldsymbol{\eta}_g \mid \boldsymbol{\eta}_a \mid \boldsymbol{\eta}_{bg} \mid \boldsymbol{\eta}_{ba})$, 각 3열. 공분산 전파 $B_k Q_d B_k^\top$에서는 부호가 $\pm$ 두 번 곱해져 사라지므로 결과 covariance는 동일.

비제로 블록 설명:

- $B_k[0{:}3, 0{:}3] = -J_r^{-1}\Delta t$: gyro 노이즈 $\rightarrow$ 회전 오차
- $B_k[3{:}6, 3{:}6] = -\Delta R_{ik}\Delta t$: accel 노이즈 $\rightarrow$ 속도 오차
- $B_k[6{:}9, 3{:}6] = -\tfrac{1}{2}\Delta R_{ik}\Delta t^2$: accel 노이즈 $\rightarrow$ 위치 오차

주의: 바이어스 노이즈는 B_k에 없다
$\boldsymbol{\eta}_{bg}, \boldsymbol{\eta}_{ba}$는 바이어스 오차 상태 $\delta b_g, \delta b_a$에 직접 들어가므로, **별도의 bias Jacobian 방정식**으로 처리한다 (다음 슬라이드).

기호 정의

$$J_{b_g}^{\Delta R_k} \triangleq \frac{\partial \Delta R_{ik}}{\partial b_g} \in \mathbb{R}^{3\times 3}$$

$\frac{\partial \Delta R_{ij}}{\partial b_g}$ Jacobian 유도 — 핵심

기호 정의

$$J_{b_g}^{\Delta R_k} \triangleq \frac{\partial \Delta R_{ik}}{\partial b_g} \in \mathbb{R}^{3\times 3}$$

$\Delta R_{i,k+1} = \Delta R_{ik}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t)$ 에서 $\tilde{\boldsymbol{\omega}}_k = \boldsymbol{\omega}_{m,k} - b_g$ 이므로 b_g에 대해 미분하면:

재귀 관계 (체인룰 + BCH 근사)

$$J_{b_g}^{\Delta R_{k+1}} = \mathrm{Exp}(-\tilde{\boldsymbol{\omega}}_k \Delta t)\; J_{b_g}^{\Delta R_k} - J_r^{-1}(\tilde{\boldsymbol{\omega}}_k \Delta t)\; \Delta t$$

$\frac{\partial \Delta R_{ij}}{\partial b_g}$ Jacobian 유도 — 핵심

기호 정의

$$J_{b_g}^{\Delta R_k} \triangleq \frac{\partial \Delta R_{ik}}{\partial b_g} \in \mathbb{R}^{3\times 3}$$

$\Delta R_{i,k+1} = \Delta R_{ik}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t)$ 에서 $\tilde{\boldsymbol{\omega}}_k = \boldsymbol{\omega}_{m,k} - b_g$ 이므로 b_g에 대해 미분하면:

재귀 관계 (체인룰 + BCH 근사)

$$J_{b_g}^{\Delta R_{k+1}} = \mathrm{Exp}(-\tilde{\boldsymbol{\omega}}_k \Delta t)\; J_{b_g}^{\Delta R_k} - J_r^{-1}(\tilde{\boldsymbol{\omega}}_k \Delta t)\; \Delta t$$

초기 조건:

$$J_{b_g}^{\Delta R_{ii}} = \mathbf{0}_{3\times 3} \quad (\text{frame } i\text{에서 자기 자신 = 항등원, 바이어스 의존 없음})$$

$\frac{\partial \Delta R_{ij}}{\partial b_g}$ Jacobian 유도 — 핵심

기호 정의

$$J_{b_g}^{\Delta R_k} \triangleq \frac{\partial \Delta R_{ik}}{\partial b_g} \in \mathbb{R}^{3\times 3}$$

$\Delta R_{i,k+1} = \Delta R_{ik}\,\mathrm{Exp}(\tilde{\omega}_k \Delta t)$ 에서 $\tilde{\omega}_k = \omega_{m,k} - b_g$ 이므로 b_g에 대해 미분하면:

재귀 관계 (체인룰 + BCH 근사)

$$J_{b_g}^{\Delta R_{k+1}} = \mathrm{Exp}(-\tilde{\omega}_k \Delta t)\; J_{b_g}^{\Delta R_k} - J_r^{-1}(\tilde{\omega}_k \Delta t)\;\Delta t$$

초기 조건:

$$J_{b_g}^{\Delta R_{ii}} = \mathbf{0}_{3\times 3} \quad \text{(frame } i\text{에서 자기 자신 = 항등원, 바이어스 의존 없음)}$$

이 Jacobian이 preintegration Jacobian의 핵심인 이유

- $J_{b_g}^{\Delta R}$는 속도/위치 Jacobian $J_{b_g}^{\Delta v}$, $J_{b_g}^{\Delta p}$의 **입력**이 됨
- 회전이 SO(3) 위에서 propagate되므로 단순한 유클리드 미분이 아님
- J_r^{-1}이 매 스텝 누적 ⇒ 정확한 manifold 상의 Jacobian

속도 재귀식 미분: $\Delta v_{i,k+1} = \Delta v_{ik} + \Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t$

$\frac{\partial \Delta v_{ij}}{\partial b_g}$, $\frac{\partial \Delta v_{ij}}{\partial b_a}$ 유도

속도 재귀식 미분: $\Delta v_{i,k+1} = \Delta v_{ik} + \Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t$

b_g에 대한 Jacobian 재귀식

$$J_{b_g}^{\Delta v_{k+1}} = J_{b_g}^{\Delta v_k} - \Delta R_{ik}\ [\tilde{\boldsymbol{a}}_k]_\times\ J_{b_g}^{\Delta R_k}\ \Delta t$$

(ΔR_{ik}도 b_g에 의존함 $\Rightarrow$ 체인룰 필요)

속도 재귀식 미분: $\Delta v_{i,k+1} = \Delta v_{ik} + \Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t$

b_g에 대한 Jacobian 재귀식

$$J_{b_g}^{\Delta v_{k+1}} = J_{b_g}^{\Delta v_k} - \Delta R_{ik}\;[\tilde{\boldsymbol{a}}_k]_\times\;J_{b_g}^{\Delta R_k}\;\Delta t$$

(ΔR_{ik}도 b_g에 의존함 $\Rightarrow$ 체인룰 필요)

b_a에 대한 Jacobian 재귀식

$$J_{b_a}^{\Delta v_{k+1}} = J_{b_a}^{\Delta v_k} - \Delta R_{ik}\;\Delta t$$

($\tilde{\boldsymbol{a}}_k = \boldsymbol{a}_{m,k} - b_a$ 이므로 $\partial \tilde{\boldsymbol{a}}_k / \partial b_a = -I$)

$\frac{\partial \Delta v_{ij}}{\partial b_g}$, $\frac{\partial \Delta v_{ij}}{\partial b_a}$ 유도

속도 재귀식 미분: $\Delta v_{i,k+1} = \Delta v_{ik} + \Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t$

b_g에 대한 Jacobian 재귀식

$$J_{b_g}^{\Delta v_{k+1}} = J_{b_g}^{\Delta v_k} - \Delta R_{ik}\,\left[\tilde{\boldsymbol{a}}_k\right]_\times\,J_{b_g}^{\Delta R_k}\,\Delta t$$

(ΔR_{ik}도 b_g에 의존함 $\Rightarrow$ 체인룰 필요)

b_a에 대한 Jacobian 재귀식

$$J_{b_a}^{\Delta v_{k+1}} = J_{b_a}^{\Delta v_k} - \Delta R_{ik}\,\Delta t$$

($\tilde{\boldsymbol{a}}_k = \boldsymbol{a}_{m,k} - b_a$ 이므로 $\partial \tilde{\boldsymbol{a}}_k / \partial b_a = -I$)

초기 조건: $J_{b_g}^{\Delta v_{ii}} = J_{b_a}^{\Delta v_{ii}} = \mathbf{0}_{3\times 3}$

$\frac{\partial \Delta v_{ij}}{\partial b_g}$, $\frac{\partial \Delta v_{ij}}{\partial b_a}$ 유도

속도 재귀식 미분: $\Delta v_{i,k+1} = \Delta v_{ik} + \Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t$

b_g에 대한 Jacobian 재귀식

$$J_{b_g}^{\Delta v_{k+1}} = J_{b_g}^{\Delta v_k} - \Delta R_{ik}\;[\tilde{\boldsymbol{a}}_k]_\times\;J_{b_g}^{\Delta R_k}\;\Delta t$$

(ΔR_{ik}도 b_g에 의존함 $\Rightarrow$ 체인룰 필요)

b_a에 대한 Jacobian 재귀식

$$J_{b_a}^{\Delta v_{k+1}} = J_{b_a}^{\Delta v_k} - \Delta R_{ik}\;\Delta t$$

($\tilde{\boldsymbol{a}}_k = \boldsymbol{a}_{m,k} - b_a$ 이므로 $\partial \tilde{\boldsymbol{a}}_k / \partial b_a = -I$)

초기 조건: $J_{b_g}^{\Delta v_{ii}} = J_{b_a}^{\Delta v_{ii}} = \mathbf{0}_{3\times 3}$

누적 합 형태로 표현:

$$J_{b_g}^{\Delta v_{ij}} = -\sum_{k=i}^{j-1} \Delta R_{ik}\;[\tilde{\boldsymbol{a}}_k]_\times\;J_{b_g}^{\Delta R_{ik}}\;\Delta t, \qquad J_{b_a}^{\Delta v_{ij}} = -\sum_{k=i}^{j-1} \Delta R_{ik}\;\Delta t$$

각각 $\mathbb{R}^{3\times 3}$.

위치 재귀식 미분: $\Delta p_{i,k+1} = \Delta p_{ik} + \Delta v_{ik}\,\Delta t + \frac{1}{2}\Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t^2$

$\frac{\partial \Delta p_{ij}}{\partial b_g}$, $\frac{\partial \Delta p_{ij}}{\partial b_a}$ 유도

위치 재귀식 미분: $\Delta p_{i,k+1} = \Delta p_{ik} + \Delta v_{ik}\,\Delta t + \frac{1}{2}\Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t^2$

b_g에 대한 Jacobian 재귀식

$$J_{b_g}^{\Delta p_{k+1}} = J_{b_g}^{\Delta p_k} + J_{b_g}^{\Delta v_k}\,\Delta t - \tfrac{1}{2}\,\Delta R_{ik}\,\left[\tilde{\boldsymbol{a}}_k\right]_\times\,J_{b_g}^{\Delta R_k}\,\Delta t^2$$

$\frac{\partial \Delta p_{ij}}{\partial b_g}$, $\frac{\partial \Delta p_{ij}}{\partial b_a}$ 유도

위치 재귀식 미분: $\Delta p_{i,k+1} = \Delta p_{ik} + \Delta v_{ik}\,\Delta t + \frac{1}{2}\Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t^2$

b_g에 대한 Jacobian 재귀식

$$J_{b_g}^{\Delta p_{k+1}} = J_{b_g}^{\Delta p_k} + J_{b_g}^{\Delta v_k}\,\Delta t - \tfrac{1}{2}\,\Delta R_{ik}\,\left[\tilde{\boldsymbol{a}}_k\right]_\times\,J_{b_g}^{\Delta R_k}\,\Delta t^2$$

b_a에 대한 Jacobian 재귀식

$$J_{b_a}^{\Delta p_{k+1}} = J_{b_a}^{\Delta p_k} + J_{b_a}^{\Delta v_k}\,\Delta t - \tfrac{1}{2}\,\Delta R_{ik}\,\Delta t^2$$

$\frac{\partial \Delta p_{ij}}{\partial b_g}$, $\frac{\partial \Delta p_{ij}}{\partial b_a}$ 유도

위치 재귀식 미분: $\Delta p_{i,k+1} = \Delta p_{ik} + \Delta v_{ik}\,\Delta t + \frac{1}{2}\Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t^2$

b_g에 대한 Jacobian 재귀식

$$J_{b_g}^{\Delta p_{k+1}} = J_{b_g}^{\Delta p_k} + J_{b_g}^{\Delta v_k}\,\Delta t - \tfrac{1}{2}\,\Delta R_{ik}\,\left[\tilde{\boldsymbol{a}}_k\right]_\times\,J_{b_g}^{\Delta R_k}\,\Delta t^2$$

b_a에 대한 Jacobian 재귀식

$$J_{b_a}^{\Delta p_{k+1}} = J_{b_a}^{\Delta p_k} + J_{b_a}^{\Delta v_k}\,\Delta t - \tfrac{1}{2}\,\Delta R_{ik}\,\Delta t^2$$

초기 조건: $J_{b_g}^{\Delta p_{ii}} = J_{b_a}^{\Delta p_{ii}} = \mathbf{0}_{3\times 3}$

$\frac{\partial \Delta p_{ij}}{\partial b_g}$, $\frac{\partial \Delta p_{ij}}{\partial b_a}$ 유도

위치 재귀식 미분: $\Delta p_{i,k+1} = \Delta p_{ik} + \Delta v_{ik}\,\Delta t + \frac{1}{2}\Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t^2$

b_g에 대한 Jacobian 재귀식

$$J_{b_g}^{\Delta p_{k+1}} = J_{b_g}^{\Delta p_k} + J_{b_g}^{\Delta v_k}\,\Delta t - \tfrac{1}{2}\,\Delta R_{ik}\,\left[\tilde{\boldsymbol{a}}_k\right]_\times\,J_{b_g}^{\Delta R_k}\,\Delta t^2$$

b_a에 대한 Jacobian 재귀식

$$J_{b_a}^{\Delta p_{k+1}} = J_{b_a}^{\Delta p_k} + J_{b_a}^{\Delta v_k}\,\Delta t - \tfrac{1}{2}\,\Delta R_{ik}\,\Delta t^2$$

초기 조건: $J_{b_g}^{\Delta p_{ii}} = J_{b_a}^{\Delta p_{ii}} = \mathbf{0}_{3\times 3}$

누적 합 형태

$$J_{b_g}^{\Delta p_{ij}} = \sum_{k=i}^{j-1}\left(J_{b_g}^{\Delta v_{ik}}\,\Delta t - \tfrac{1}{2}\Delta R_{ik}\,[\tilde{\boldsymbol{a}}_k]_\times\,J_{b_g}^{\Delta R_{ik}}\,\Delta t^2\right)$$

$$J_{b_a}^{\Delta p_{ij}} = -\tfrac{1}{2}\sum_{k=i}^{j-1}\Delta R_{ik}\,\Delta t^2$$

이로써 총 **6개**의 Bias Jacobian 행렬이 완성됨.

바이어스 변화 시 1차 보정 공식

ESKF update 후 바이어스 추정치가 $b_g \to b_g + \delta b_g$, $b_a \to b_a + \delta b_a$ 로 바뀌었다 하자.

바이어스 변화 시 1차 보정 공식

ESKF update 후 바이어스 추정치가 $b_g \to b_g + \delta b_g$, $b_a \to b_a + \delta b_a$ 로 바뀌었다 하자.

Preintegration 측정값 선형 보정 (재적분 없이!)

$$\Delta\hat{R}_{ij} \approx \Delta R_{ij}\ \mathrm{Exp}\left(J_{b_g}^{\Delta R_{ij}}\,\delta b_g\right) \in \mathrm{SO}(3)$$

$$\Delta\hat{v}_{ij} \approx \Delta v_{ij} + J_{b_g}^{\Delta v_{ij}}\,\delta b_g + J_{b_a}^{\Delta v_{ij}}\,\delta b_a \in \mathbb{R}^3$$

$$\Delta\hat{p}_{ij} \approx \Delta p_{ij} + J_{b_g}^{\Delta p_{ij}}\,\delta b_g + J_{b_a}^{\Delta p_{ij}}\,\delta b_a \in \mathbb{R}^3$$

바이어스 변화 시 1차 보정 공식

ESKF update 후 바이어스 추정치가 $b_g \to b_g + \delta b_g$, $b_a \to b_a + \delta b_a$ 로 바뀌었다 하자.

Preintegration 측정값 선형 보정 (재적분 없이!)

$$\Delta\hat{R}_{ij} \approx \Delta R_{ij}\ \mathrm{Exp}\big(J_{b_g}^{\Delta R_{ij}}\,\delta b_g\big) \ \in\ \mathrm{SO}(3)$$

$$\Delta\hat{v}_{ij} \approx \Delta v_{ij} + J_{b_g}^{\Delta v_{ij}}\,\delta b_g + J_{b_a}^{\Delta v_{ij}}\,\delta b_a \ \in\ \mathbb{R}^3$$

$$\Delta\hat{p}_{ij} \approx \Delta p_{ij} + J_{b_g}^{\Delta p_{ij}}\,\delta b_g + J_{b_a}^{\Delta p_{ij}}\,\delta b_a \ \in\ \mathbb{R}^3$$

계산 절감

- 재적분: $O(N)$ matrix multiply $\times$ 업데이트 횟수
- 선형 보정: $O(1)$ 행렬-벡터 곱 3번
- Factor graph에서 매우 유용: bias 업데이트마다 호출

바이어스 변화 시 1차 보정 공식

ESKF update 후 바이어스 추정치가 $b_g \to b_g + \delta b_g$, $b_a \to b_a + \delta b_a$ 로 바뀌었다 하자.

Preintegration 측정값 선형 보정 (재적분 없이!)

$$\Delta\hat{R}_{ij} \approx \Delta R_{ij}\ \mathrm{Exp}\left(J_{b_g}^{\Delta R_{ij}}\,\delta b_g\right) \in \mathrm{SO}(3)$$

$$\Delta\hat{v}_{ij} \approx \Delta v_{ij} + J_{b_g}^{\Delta v_{ij}}\,\delta b_g + J_{b_a}^{\Delta v_{ij}}\,\delta b_a \in \mathbb{R}^3$$

$$\Delta\hat{p}_{ij} \approx \Delta p_{ij} + J_{b_g}^{\Delta p_{ij}}\,\delta b_g + J_{b_a}^{\Delta p_{ij}}\,\delta b_a \in \mathbb{R}^3$$

계산 절감

- 재적분: $O(N)$ matrix multiply $\times$ 업데이트 횟수
- 선형 보정: $O(1)$ 행렬-벡터 곱 3번
- Factor graph에서 매우 유용: bias 업데이트마다 호출

수식 출처: Forster et al., "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry"

언제 선형 보정으로 충분한가

유효 조건
$\|\delta b_g\|$와 $\|\delta b_a\|$가 **충분히 작을 때** (Taylor 나머지항이 무시 가능)

구체적으로:

$$\|\delta b_g\| \ll \sigma_{bg}\sqrt{T}, \qquad \|\delta b_a\| \ll \sigma_{ba}\sqrt{T}$$

(T: preintegration 구간 길이)

언제 선형 보정으로 충분한가

유효 조건
$\|\delta b_g\|$와 $\|\delta b_a\|$가 **충분히 작을 때** (Taylor 나머지항이 무시 가능)

구체적으로:

$$\|\delta b_g\| \ll \sigma_{bg}\sqrt{T}, \qquad \|\delta b_a\| \ll \sigma_{ba}\sqrt{T}$$

(T: preintegration 구간 길이)

선형 보정으로 충분한 경우

- ESKF 업데이트 직후
  (bias 변화량 작음)
- 짧은 preintegration 구간
- 느린 운동, 낮은 바이어스 드리프트

재적분이 필요한 경우

- 바이어스 업데이트량이 threshold 초과
- 긴 구간 (수 초 이상)
- 격렬한 회전 운동

언제 선형 보정으로 충분한가

유효 조건
$\|\delta b_g\|$와 $\|\delta b_a\|$가 **충분히 작을 때** (Taylor 나머지항이 무시 가능)

구체적으로:

$$\|\delta b_g\| \ll \sigma_{bg}\sqrt{T}, \qquad \|\delta b_a\| \ll \sigma_{ba}\sqrt{T}$$

(T: preintegration 구간 길이)

선형 보정으로 충분한 경우

- ESKF 업데이트 직후
  (bias 변화량 작음)
- 짧은 preintegration 구간
- 느린 운동, 낮은 바이어스 드리프트

재적분이 필요한 경우

- 바이어스 업데이트량이 threshold 초과
- 긴 구간 (수 초 이상)
- 격렬한 회전 운동

OpenVINS 구현 전략
`Propagator::repropagate()` vs `correctPropagation()`
바이어스 변화량이 threshold 미만이면 보정, 초과하면 재전파.
`options.dt_slam_delay`와 연동된 성능-정확도 trade-off.

Preintegration Factor — 잔차 형태 (Factor Graph 연결)

상태: $\pi_i = (R_i, v_i, p_i)$, $\pi_j = (R_j, v_j, p_j)$, 바이어스 (b_g, b_a).

Preintegration Factor — 잔차 형태 (Factor Graph 연결)

상태: $\pi_i = (R_i, v_i, p_i)$, $\pi_j = (R_j, v_j, p_j)$, 바이어스 (b_g, b_a).

잔차 벡터 $\boldsymbol{r}_{ij} \in \mathbb{R}^9$

$$r_{\Delta R} = \mathsf{Log}\Big(\Delta\hat{R}_{ij}^{\top}\, R_i^{\top}\, R_j\Big) \;\in\; \mathbb{R}^3$$

$$r_{\Delta v} = R_i^{\top}\big(v_j - v_i - \boldsymbol{g}\,\Delta t_{ij}\big) - \Delta\hat{v}_{ij} \;\in\; \mathbb{R}^3$$

$$r_{\Delta p} = R_i^{\top}\Big(p_j - p_i - v_i\,\Delta t_{ij} - \tfrac{1}{2}\boldsymbol{g}\,\Delta t_{ij}^2\Big) - \Delta\hat{p}_{ij} \;\in\; \mathbb{R}^3$$

Preintegration Factor — 잔차 형태 (Factor Graph 연결)

상태: $\pi_i = (R_i, v_i, p_i)$, $\pi_j = (R_j, v_j, p_j)$, 바이어스 (b_g, b_a).

잔차 벡터 $\boldsymbol{r}_{ij} \in \mathbb{R}^9$

$$r_{\Delta R} = \mathsf{Log}\Big(\Delta\hat{R}_{ij}^{\top}\, R_i^{\top}\, R_j\Big) \;\in\; \mathbb{R}^3$$

$$r_{\Delta v} = R_i^{\top}\big(v_j - v_i - \boldsymbol{g}\,\Delta t_{ij}\big) - \Delta\hat{v}_{ij} \;\in\; \mathbb{R}^3$$

$$r_{\Delta p} = R_i^{\top}\Big(p_j - p_i - v_i\,\Delta t_{ij} - \tfrac{1}{2}\boldsymbol{g}\,\Delta t_{ij}^2\Big) - \Delta\hat{p}_{ij} \;\in\; \mathbb{R}^3$$

여기서 $\Delta\hat{R}_{ij}$, $\Delta\hat{v}_{ij}$, $\Delta\hat{p}_{ij}$는 현재 바이어스에서 **선형 보정된** 값.

Preintegration Factor — 잔차 형태 (Factor Graph 연결)

상태: $\pi_i = (R_i, v_i, p_i)$, $\pi_j = (R_j, v_j, p_j)$, 바이어스 (b_g, b_a).

잔차 벡터 $\boldsymbol{r}_{ij} \in \mathbb{R}^9$

$$r_{\Delta R} = \mathrm{Log}\Big(\Delta\hat{R}_{ij}^{\top}\, R_i^{\top}\, R_j\Big) \in \mathbb{R}^3$$

$$r_{\Delta v} = R_i^{\top}\big(v_j - v_i - \boldsymbol{g}\,\Delta t_{ij}\big) - \Delta\hat{v}_{ij} \in \mathbb{R}^3$$

$$r_{\Delta p} = R_i^{\top}\Big(p_j - p_i - v_i\,\Delta t_{ij} - \tfrac{1}{2}\boldsymbol{g}\,\Delta t_{ij}^2\Big) - \Delta\hat{p}_{ij} \in \mathbb{R}^3$$

여기서 $\Delta\hat{R}_{ij}$, $\Delta\hat{v}_{ij}$, $\Delta\hat{p}_{ij}$는 현재 바이어스에서 **선형 보정된** 값.

Factor Graph에서의 역할

- 잔차 $\boldsymbol{r}_{ij}$를 최소화하는 non-linear least squares
- $\mathcal{I}_{ij} = \Sigma_{ij}^{-1}$: 정보 행렬 (preintegration 공분산의 역)
- 비용함수: $\boldsymbol{r}_{ij}^{\top}\mathcal{I}_{ij}\boldsymbol{r}_{ij}$
- 이 잔차의 Jacobian에 $J_{b_g}^{\Delta R}$ 등이 등장 (다음 슬라이드)

Preintegration Factor Jacobian — $\partial \boldsymbol{r} / \partial$state

전체 Jacobian: $\boldsymbol{J} \in \mathbb{R}^{9 \times 24}$ (9 residuals $\times$ 24 = state$_i$ 9 + state$_j$ 9 + bias$_i$ 6)

Preintegration Factor Jacobian — $\partial r / \partial$ state

전체 Jacobian: $\boldsymbol{J} \in \mathbb{R}^{9\times 24}$ (9 residuals × 24 = state$_i$ 9 + state$_j$ 9 + bias$_i$ 6)

$\partial r_{\Delta R}/\partial$**state**

$$\frac{\partial r_{\Delta R}}{\partial \delta\phi_i} = -\left(\Delta\hat{R}_{ij}^{\top} R_i^{\top} R_j\right)^{-\top}, \quad \frac{\partial r_{\Delta R}}{\partial \delta\phi_j} = J_r^{-1}\Big(\mathrm{Log}(\Delta\hat{R}_{ij}^{\top} R_i^{\top} R_j)\Big), \quad \frac{\partial r_{\Delta R}}{\partial \delta b_g} = -J_r^{-1}(\cdot)\, J_{b_g}^{\Delta R_{ij}}$$

Preintegration Factor Jacobian — $\partial \boldsymbol{r} / \partial$ state

전체 Jacobian: $\boldsymbol{J} \in \mathbb{R}^{9\times 24}$ (9 residuals × 24 = state$_i$ 9 + state$_j$ 9 + bias$_i$ 6)

$\partial r_{\Delta R} / \partial$ **state**

$$\frac{\partial r_{\Delta R}}{\partial \delta \phi_i} = -\left(\Delta \hat{R}_{ij}^{\top} R_i^{\top} R_j\right)^{-\top}, \quad \frac{\partial r_{\Delta R}}{\partial \delta \phi_j} = J_r^{-1}\left(\mathrm{Log}(\Delta \hat{R}_{ij}^{\top} R_i^{\top} R_j)\right), \quad \frac{\partial r_{\Delta R}}{\partial \delta b_g} = -J_r^{-1}(\cdot)\, J_{b_g}^{\Delta R_{ij}}$$

$\partial r_{\Delta v} / \partial$ **state (대표 항)**

$$\frac{\partial r_{\Delta v}}{\partial \delta \phi_i} = \left[R_i^{\top}(v_j - v_i - \boldsymbol{g}\Delta t_{ij})\right]_{\times}$$

$$\frac{\partial r_{\Delta v}}{\partial \delta b_g} = -J_{b_g}^{\Delta v_{ij}}, \quad \frac{\partial r_{\Delta v}}{\partial \delta b_a} = -J_{b_a}^{\Delta v_{ij}}$$

Preintegration Factor Jacobian — $\partial r / \partial$ state

전체 Jacobian: $\boldsymbol{J} \in \mathbb{R}^{9\times 24}$ (9 residuals $\times$ 24 = state$_i$ 9 + state$_j$ 9 + bias$_i$ 6)

$\partial r_{\Delta R}/\partial$**state**

$$\frac{\partial r_{\Delta R}}{\partial \delta\phi_i} = -\left(\Delta\hat{R}_{ij}^{\top} R_i^{\top} R_j\right)^{-\top}, \quad \frac{\partial r_{\Delta R}}{\partial \delta\phi_j} = J_r^{-1}\Big(\mathrm{Log}(\Delta\hat{R}_{ij}^{\top} R_i^{\top} R_j)\Big), \quad \frac{\partial r_{\Delta R}}{\partial \delta b_g} = -J_r^{-1}(\cdot)\, J_{b_g}^{\Delta R_{ij}}$$

$\partial r_{\Delta v}/\partial$**state (대표 항)**

$$\frac{\partial r_{\Delta v}}{\partial \delta\phi_i} = \Big[R_i^{\top}(v_j - v_i - \boldsymbol{g}\Delta t_{ij})\Big]_{\times}$$

$$\frac{\partial r_{\Delta v}}{\partial \delta b_g} = -J_{b_g}^{\Delta v_{ij}}, \quad \frac{\partial r_{\Delta v}}{\partial \delta b_a} = -J_{b_a}^{\Delta v_{ij}}$$

$\partial r_{\Delta p}/\partial$**state (bias 항)**

$$\frac{\partial r_{\Delta p}}{\partial \delta b_g} = -J_{b_g}^{\Delta p_{ij}}, \quad \frac{\partial r_{\Delta p}}{\partial \delta b_a} = -J_{b_a}^{\Delta p_{ij}}$$

Preintegration Factor Jacobian — $\partial r / \partial$ state

전체 Jacobian: $\boldsymbol{J} \in \mathbb{R}^{9\times 24}$ (9 residuals × 24 = state$_i$ 9 + state$_j$ 9 + bias$_i$ 6)

$\partial r_{\Delta R} / \partial$ **state**

$$\frac{\partial r_{\Delta R}}{\partial \delta\phi_i} = -\left(\Delta\hat{R}_{ij}^{\top} R_i^{\top} R_j\right)^{-\top}, \quad \frac{\partial r_{\Delta R}}{\partial \delta\phi_j} = J_r^{-1}\left(\mathrm{Log}(\Delta\hat{R}_{ij}^{\top} R_i^{\top} R_j)\right), \quad \frac{\partial r_{\Delta R}}{\partial \delta b_g} = -J_r^{-1}(\cdot)\, J_{b_g}^{\Delta R_{ij}}$$

$\partial r_{\Delta v} / \partial$ **state (대표 항)**

$$\frac{\partial r_{\Delta v}}{\partial \delta\phi_i} = \left[R_i^{\top}(v_j - v_i - \boldsymbol{g}\Delta t_{ij})\right]_{\times}$$

$$\frac{\partial r_{\Delta v}}{\partial \delta b_g} = -J_{b_g}^{\Delta v_{ij}}, \quad \frac{\partial r_{\Delta v}}{\partial \delta b_a} = -J_{b_a}^{\Delta v_{ij}}$$

$\partial r_{\Delta p} / \partial$ **state (bias 항)**

$$\frac{\partial r_{\Delta p}}{\partial \delta b_g} = -J_{b_g}^{\Delta p_{ij}}, \quad \frac{\partial r_{\Delta p}}{\partial \delta b_a} = -J_{b_a}^{\Delta p_{ij}}$$

결론

앞서 유도한 6개의 Bias Jacobian이 모두 Factor Jacobian에 직접 등장한다.

OpenVINS propagate() 코드 — 공분산 전파 부분

Listing 1: Propagator.cpp (OpenVINS, GPLv3, 개략화)

```
// --- 1. 이산 A_k, B_k 구성 ---
Eigen::Matrix3d dR = Exp(omegatil * dt);         // Delta R_{i,k+1}
Eigen::Matrix3d Jr_inv = rightJacobianInverse(omegatil * dt);

// A_k (9x9 블록행렬 )
Phi.block<3,3>(0,0) = dR.transpose();            // Exp(-omegatil*dt)
Phi.block<3,3>(3,0) = -dR_accum * skew(atil) * dt;
Phi.block<3,3>(6,0) = -0.5 * dR_accum * skew(atil) * dt * dt;
Phi.block<3,3>(3,3) = Eigen::Matrix3d::Identity();
Phi.block<3,3>(6,3) = Eigen::Matrix3d::Identity() * dt;
Phi.block<3,3>(6,6) = Eigen::Matrix3d::Identity();

// B_k (9x6: gyro noise | accel noise)
G.block<3,3>(0,0) = Jr_inv * dt;                 // dphi / d eta_g
G.block<3,3>(3,3) = dR_accum * dt;               // dv   / d eta_a
G.block<3,3>(6,3) = 0.5 * dR_accum * dt * dt;  // dp   / d eta_a

// --- 2. 공분산전파  ---
Eigen::MatrixXd Qd = G * Qc * G.transpose();    // B_k * Q_d * B_k^T
P_new = Phi * P * Phi.transpose() + Qd;

// --- 3. Preintegration Bias Jacobians 누적() ---
J_R  = dR.transpose() * J_R  - Jr_inv * dt;     // partial DeltaR / partial bg
J_v += -dR_accum * skew(atil) * J_R * dt;       // partial Deltav / partial bg
J_va = -dR_accum * dt;                           // partial Deltav / partial ba  (accumulate)
J_p += J_v_prev * dt - 0.5*dR_accum*skew(atil)*J_R*dt*dt;
J_pa -= 0.5 * dR_accum * dt * dt;                // partial Deltap / partial ba
```

Source: ov_msckf/src/state/Propagator.cpp, github.com/rpng/open_vins (GPLv3)

GTSAM ImuFactor 연결 — Forster 2017 직접 구현

GTSAM의 ImuFactor는 Forster 2017의 on-manifold preintegration을 그대로 구현.

Listing 2: GTSAM ImuFactor 사용 예시

```
#include <gtsam/navigation/ImuFactor.h>
#include <gtsam/navigation/PreintegrationParams.h>

// --- IMU 파라미터설정 ---
auto params = gtsam::PreintegrationParams::MakeSharedU(9.81); // NED convention
params->gyroscopeCovariance    = sigma_g * sigma_g * gtsam::I_3x3;
params->accelerometerCovariance = sigma_a * sigma_a * gtsam::I_3x3;
params->integrationCovariance = 1e-8 * gtsam::I_3x3;

gtsam::imuBias::ConstantBias prior_bias(accel_bias, gyro_bias);
gtsam::PreintegratedImuMeasurements pim(params, prior_bias);

// --- IMU 측정값적분 ---
for (auto& [dt, accel, gyro] : imu_data) {
    pim.integrateMeasurement(accel, gyro, dt);
    // 내부에서 DeltaR, Deltav, Deltap + A_k, B_k + 6 Jacobians 동시업데이트
}

// --- Factor 추가 ---
graph.add(gtsam::ImuFactor(
    X(i), V(i), X(j), V(j), B(i), pim
)); // pim 안에 Sigma_ij, J_bias 모두포함
```

GTSAM ImuFactor 연결 — Forster 2017 직접 구현

GTSAM의 ImuFactor는 Forster 2017의 on-manifold preintegration을 그대로 구현.

Listing 3: GTSAM ImuFactor 사용 예시

```
#include <gtsam/navigation/ImuFactor.h>
#include <gtsam/navigation/PreintegrationParams.h>

// --- IMU 파라미터설정  ---
auto params = gtsam::PreintegrationParams::MakeSharedU(9.81); // NED convention
params->gyroscopeCovariance   = sigma_g * sigma_g * gtsam::I_3x3;
params->accelerometerCovariance = sigma_a * sigma_a * gtsam::I_3x3;
params->integrationCovariance = 1e-8 * gtsam::I_3x3;

gtsam::imuBias::ConstantBias prior_bias(accel_bias, gyro_bias);
gtsam::PreintegratedImuMeasurements pim(params, prior_bias);

// --- IMU 측정값적분  ---
for (auto& [dt, accel, gyro] : imu_data) {
    pim.integrateMeasurement(accel, gyro, dt);
    // 내부에서 DeltaR, Deltav, Deltap + A_k, B_k + 6 Jacobians 동시업데이트
}

// --- Factor 추가 ---
graph.add(gtsam::ImuFactor(
    X(i), V(i), X(j), V(j), B(i), pim
)); // pim 안에 Sigma_ij, J_bias 모두포함
```

내부 구현과의 대응

PreintegratedRotation::update() $\leftrightarrow J_{b_g}^{\Delta R}$ 재귀

ManifoldPreintegration::update() $\leftrightarrow A_k, B_k, \Sigma_{ij}$ 전파

Noise Covariance Σ_{ij} 최종 형태

재귀 전파를 $k = i$ 에서 $j-1$까지 수행한 결과:

$$\Sigma_{ij} \;=\; \sum_{k=i}^{j-1} \left(\prod_{l=k+1}^{j-1} A_l \right) B_k \, Q_d \, B_k^{\top} \left(\prod_{l=k+1}^{j-1} A_l \right)^{\top} \;\in\; \mathbb{R}^{9\times 9}$$

(재귀로 $O(N)$에 계산)

Noise Covariance Σ_{ij} 최종 형태

재귀 전파를 $k = i$ 에서 $j-1$까지 수행한 결과:

$$\Sigma_{ij} \;=\; \sum_{k=i}^{j-1} \left(\prod_{l=k+1}^{j-1} A_l \right) B_k \, Q_d \, B_k^{\top} \left(\prod_{l=k+1}^{j-1} A_l \right)^{\top} \;\in\; \mathbb{R}^{9\times 9}$$

(재귀로 $O(N)$에 계산)

블록 구조 (Forster 2017 eq. 33)

$$\Sigma_{ij} = \begin{pmatrix} \Sigma_{\phi\phi} & \Sigma_{\phi v} & \Sigma_{\phi p} \\ \Sigma_{v\phi} & \Sigma_{vv} & \Sigma_{vp} \\ \Sigma_{p\phi} & \Sigma_{pv} & \Sigma_{pp} \end{pmatrix} \in \mathbb{R}^{9\times 9}$$

교차항 $\Sigma_{\phi v}$, $\Sigma_{\phi p}$ 등이 회전-병진 상관을 포착.

Noise Covariance Σ_{ij} 최종 형태

재귀 전파를 $k = i$ 에서 $j-1$까지 수행한 결과:

$$\Sigma_{ij} = \sum_{k=i}^{j-1} \left(\prod_{l=k+1}^{j-1} A_l \right) B_k \, Q_d \, B_k^\top \left(\prod_{l=k+1}^{j-1} A_l \right)^\top \in \mathbb{R}^{9\times 9}$$

(재귀로 $O(N)$에 계산)

블록 구조 (Forster 2017 eq. 33)

$$\Sigma_{ij} = \begin{pmatrix} \Sigma_{\phi\phi} & \Sigma_{\phi v} & \Sigma_{\phi p} \\ \Sigma_{v\phi} & \Sigma_{vv} & \Sigma_{vp} \\ \Sigma_{p\phi} & \Sigma_{pv} & \Sigma_{pp} \end{pmatrix} \in \mathbb{R}^{9\times 9}$$

교차항 $\Sigma_{\phi v}$, $\Sigma_{\phi p}$ 등이 회전-병진 상관을 포착.

ESKF에서의 역할
Σ_{ij}는 IMU predict 후
ESKF 공분산 P의 일부가 됨
(measurement 업데이트 전)

Factor Graph에서의 역할
$\mathcal{I}_{ij} = \Sigma_{ij}^{-1}$
= IMU factor의 **정보 행렬**
잔차 가중치로 사용

Noise Covariance Σ_{ij} 최종 형태

재귀 전파를 $k = i$ 에서 $j-1$까지 수행한 결과:

$$\Sigma_{ij} = \sum_{k=i}^{j-1} \left(\prod_{l=k+1}^{j-1} A_l \right) B_k \, Q_d \, B_k^\top \left(\prod_{l=k+1}^{j-1} A_l \right)^\top \in \mathbb{R}^{9\times 9}$$

(재귀로 $O(N)$에 계산)

블록 구조 (Forster 2017 eq. 33)

$$\Sigma_{ij} = \begin{pmatrix} \Sigma_{\phi\phi} & \Sigma_{\phi v} & \Sigma_{\phi p} \\ \Sigma_{v\phi} & \Sigma_{vv} & \Sigma_{vp} \\ \Sigma_{p\phi} & \Sigma_{pv} & \Sigma_{pp} \end{pmatrix} \in \mathbb{R}^{9\times 9}$$

교차항 $\Sigma_{\phi v}$, $\Sigma_{\phi p}$ 등이 회전-병진 상관을 포착.

ESKF에서의 역할
Σ_{ij}는 IMU predict 후
ESKF 공분산 P의 일부가 됨
(measurement 업데이트 전)

Factor Graph에서의 역할
$\mathcal{I}_{ij} = \Sigma_{ij}^{-1}$
= IMU factor의 **정보 행렬**
잔차 가중치로 사용

바이어스 불확실도 $\Sigma_b \in \mathbb{R}^{6\times 6}$는 별도의 bias 랜덤 보행 공분산으로 전파.

Forster 2017 vs 이전 방식 비교

Lupton & Sukkarieh (2012)

- 유클리드 공간에서 적분
- 회전 오차를 $\mathbb{R}^3$로 근사 (소각 가정)
- Noise propagation: 단순 연쇄 법칙
- 큰 회전에서 공분산 불일치
- 바이어스 Jacobian: 유클리드 편미분

Forster 2017 (On-Manifold)

- $\mathrm{SO}(3)$ 위에서 정확한 적분
- Exp/Log, Right Jacobian 사용
- Noise propagation: A_k, B_k 행렬
- 큰 회전에서도 일관된 공분산
- 바이어스 Jacobian: manifold 상의 정확한 편미분

Forster 2017 vs 이전 방식 비교

Lupton & Sukkarieh (2012)

- 유클리드 공간에서 적분
- 회전 오차를 $\mathbb{R}^3$로 근사 (소각 가정)
- Noise propagation: 단순 연쇄 법칙
- 큰 회전에서 공분산 불일치
- 바이어스 Jacobian: 유클리드 편미분

Forster 2017 (On-Manifold)

- $\mathrm{SO}(3)$ 위에서 정확한 적분
- Exp/Log, Right Jacobian 사용
- Noise propagation: A_k, B_k 행렬
- 큰 회전에서도 일관된 공분산
- 바이어스 Jacobian: manifold 상의 정확한 편미분

핵심 차이: SO(3) 위에서의 정확한 Noise Propagation

회전 합성 $R_1 R_2$의 불확실도는 유클리드 합과 다름.

$\mathrm{Exp}(\phi_1)\mathrm{Exp}(\phi_2) \neq \mathrm{Exp}(\phi_1 + \phi_2)$ (BCH 공식)

$\Rightarrow$ Forster 방식이 더 일관된 공분산, factor graph에서 더 정확한 추정.

Forster 2017 vs 이전 방식 비교

Lupton & Sukkarieh (2012)

- 유클리드 공간에서 적분
- 회전 오차를 $\mathbb{R}^3$로 근사 (소각 가정)
- Noise propagation: 단순 연쇄 법칙
- 큰 회전에서 공분산 불일치
- 바이어스 Jacobian: 유클리드 편미분

Forster 2017 (On-Manifold)

- $\mathrm{SO}(3)$ 위에서 정확한 적분
- Exp/Log, Right Jacobian 사용
- Noise propagation: A_k, B_k 행렬
- 큰 회전에서도 일관된 공분산
- 바이어스 Jacobian: manifold 상의 정확한 편미분

핵심 차이: SO(3) 위에서의 정확한 Noise Propagation

회전 합성 $R_1 R_2$의 불확실도는 유클리드 합과 다름.

$\mathrm{Exp}(\phi_1)\mathrm{Exp}(\phi_2) \neq \mathrm{Exp}(\phi_1 + \phi_2)$ (BCH 공식)

$\Rightarrow$ Forster 방식이 더 일관된 공분산, factor graph에서 더 정확한 추정.

실험 결과 (Forster 2017 Table I): EuRoC MAV 데이터셋에서
Forster 방식이 Lupton 대비 ATE 약 15%$\sim$30% 개선.

요약: Preintegration Bias Jacobian 6개 행렬

	b_g (자이로 바이어스)	b_a (가속 바이어스)
ΔR_{ij}	$J_{b_g}^{\Delta R} \in \mathbb{R}^{3\times 3}$ (재귀)	$-$ (의존 없음)
Δv_{ij}	$J_{b_g}^{\Delta v} \in \mathbb{R}^{3\times 3}$	$J_{b_a}^{\Delta v} \in \mathbb{R}^{3\times 3}$
Δp_{ij}	$J_{b_g}^{\Delta p} \in \mathbb{R}^{3\times 3}$	$J_{b_a}^{\Delta p} \in \mathbb{R}^{3\times 3}$

요약: Preintegration Bias Jacobian 6개 행렬

	b_g (자이로 바이어스)	b_a (가속 바이어스)
ΔR_{ij}	$J_{b_g}^{\Delta R} \in \mathbb{R}^{3\times 3}$ (재귀)	– (의존 없음)
Δv_{ij}	$J_{b_g}^{\Delta v} \in \mathbb{R}^{3\times 3}$	$J_{b_a}^{\Delta v} \in \mathbb{R}^{3\times 3}$
Δp_{ij}	$J_{b_g}^{\Delta p} \in \mathbb{R}^{3\times 3}$	$J_{b_a}^{\Delta p} \in \mathbb{R}^{3\times 3}$

계산 방법

- 재귀적으로 **적분과 동시에** 계산
- $J_{b_g}^{\Delta R}$ 먼저 계산
  $\Rightarrow J_{b_g}^{\Delta v}, J_{b_g}^{\Delta p}$에 입력
- 총 추가 비용: $O(N)$ 행렬 연산

활용처

- 바이어스 업데이트 후 선형 보정
- Factor Graph 잔차 Jacobian
- ESKF 측정 업데이트 시 $\boldsymbol{H}$ 행렬

요약: Preintegration Bias Jacobian 6개 행렬

	b_g (자이로 바이어스)	b_a (가속 바이어스)
ΔR_{ij}	$J_{b_g}^{\Delta R} \in \mathbb{R}^{3\times 3}$ (재귀)	$-$ (의존 없음)
Δv_{ij}	$J_{b_g}^{\Delta v} \in \mathbb{R}^{3\times 3}$	$J_{b_a}^{\Delta v} \in \mathbb{R}^{3\times 3}$
Δp_{ij}	$J_{b_g}^{\Delta p} \in \mathbb{R}^{3\times 3}$	$J_{b_a}^{\Delta p} \in \mathbb{R}^{3\times 3}$

계산 방법

- 재귀적으로 **적분과 동시에** 계산
- $J_{b_g}^{\Delta R}$ 먼저 계산
  $\Rightarrow J_{b_g}^{\Delta v}, J_{b_g}^{\Delta p}$에 입력
- 총 추가 비용: $O(N)$ 행렬 연산

활용처

- 바이어스 업데이트 후 선형 보정
- Factor Graph 잔차 Jacobian
- ESKF 측정 업데이트 시 $\boldsymbol{H}$ 행렬

ΔR의 b_a 의존성은 0: 회전은 각속도만으로 결정되므로 가속 바이어스 무관.

총 비자명 원소: $5 \times 9 = 45$**개**

Right Jacobian J_r 복습 — 왜 preintegration에 등장하는가

Right Jacobian of SO(3)
$\boldsymbol{\phi} \in \mathbb{R}^3$, $\theta = \|\boldsymbol{\phi}\|$에 대해:

$$J_r(\boldsymbol{\phi}) = I - \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\phi}]_\times + \frac{\theta - \sin\theta}{\theta^3}[\boldsymbol{\phi}]_\times^2$$

역행렬:

$$J_r^{-1}(\boldsymbol{\phi}) = I + \frac{1}{2}[\boldsymbol{\phi}]_\times + \left(\frac{1}{\theta^2} - \frac{1+\cos\theta}{2\theta\sin\theta}\right)[\boldsymbol{\phi}]_\times^2$$

Right Jacobian J_r 복습 — 왜 preintegration에 등장하는가

Right Jacobian of SO(3)
$\boldsymbol{\phi} \in \mathbb{R}^3$, $\theta = \|\boldsymbol{\phi}\|$에 대해:

$$J_r(\boldsymbol{\phi}) = I - \frac{1-\cos\theta}{\theta^2}\,[\boldsymbol{\phi}]_\times + \frac{\theta-\sin\theta}{\theta^3}\,[\boldsymbol{\phi}]_\times^2$$

역행렬:

$$J_r^{-1}(\boldsymbol{\phi}) = I + \frac{1}{2}\,[\boldsymbol{\phi}]_\times + \left(\frac{1}{\theta^2} - \frac{1+\cos\theta}{2\theta\sin\theta}\right)[\boldsymbol{\phi}]_\times^2$$

작은 각도 근사 ($\theta \to 0$):

$$J_r(\boldsymbol{\phi}) \approx I - \tfrac{1}{2}\,[\boldsymbol{\phi}]_\times\,, \qquad J_r^{-1}(\boldsymbol{\phi}) \approx I + \tfrac{1}{2}\,[\boldsymbol{\phi}]_\times$$

Right Jacobian J_r 복습 — 왜 preintegration에 등장하는가

Right Jacobian of SO(3)
$\boldsymbol{\phi} \in \mathbb{R}^3$, $\theta = \|\boldsymbol{\phi}\|$에 대해:

$$J_r(\boldsymbol{\phi}) = I - \frac{1-\cos\theta}{\theta^2}\,[\boldsymbol{\phi}]_\times + \frac{\theta - \sin\theta}{\theta^3}\,[\boldsymbol{\phi}]_\times^2$$

역행렬:

$$J_r^{-1}(\boldsymbol{\phi}) = I + \frac{1}{2}\,[\boldsymbol{\phi}]_\times + \left(\frac{1}{\theta^2} - \frac{1+\cos\theta}{2\theta\sin\theta}\right)[\boldsymbol{\phi}]_\times^2$$

작은 각도 근사 ($\theta \to 0$):

$$J_r(\boldsymbol{\phi}) \approx I - \tfrac{1}{2}\,[\boldsymbol{\phi}]_\times\,, \qquad J_r^{-1}(\boldsymbol{\phi}) \approx I + \tfrac{1}{2}\,[\boldsymbol{\phi}]_\times$$

Preintegration에서의 역할
$\mathrm{Exp}(\boldsymbol{\phi} + \delta\boldsymbol{\phi}) \approx \mathrm{Exp}(\boldsymbol{\phi})\,\mathrm{Exp}(J_r(\boldsymbol{\phi})\,\delta\boldsymbol{\phi})$
$\Rightarrow$ gyro noise $\boldsymbol{\eta}_g$가 회전 오차 $\delta\boldsymbol{\phi}$에 미치는 영향을 J_r^{-1}이 보정

BCH 공식과 회전 오차 전파 — 상세 유도

목표: 다음을 선형화

$$\Delta R_{i,k+1} = \underbrace{\Delta R_{ik}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t)}_{\text{명목}} \quad \leftrightarrow \quad \Delta \tilde{R}_{i,k+1} = \Delta \tilde{R}_{ik}\,\mathrm{Exp}\big((\tilde{\boldsymbol{\omega}}_k - \boldsymbol{\eta}_g)\Delta t\big)$$

BCH 공식과 회전 오차 전파 — 상세 유도

목표: 다음을 선형화

$$\Delta R_{i,k+1} = \underbrace{\Delta R_{ik}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t)}_{\text{명목}} \quad \leftrightarrow \quad \Delta \tilde{R}_{i,k+1} = \Delta \tilde{R}_{ik}\,\mathrm{Exp}\big((\tilde{\boldsymbol{\omega}}_k - \boldsymbol{\eta}_g)\Delta t\big)$$

오차 정의: $\Delta \tilde{R}_{ik} = \Delta R_{ik}\,\mathrm{Exp}(\delta\phi_k)$ (left perturbation 사용 시):

$$\Delta \tilde{R}_{i,k+1} = \Delta R_{ik}\,\mathrm{Exp}(\delta\phi_k)\,\mathrm{Exp}\big((\tilde{\boldsymbol{\omega}}_k - \boldsymbol{\eta}_g)\Delta t\big)$$

BCH 공식과 회전 오차 전파 — 상세 유도

목표: 다음을 선형화

$$\Delta R_{i,k+1} = \underbrace{\Delta R_{ik}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t)}_{\text{명목}} \quad \leftrightarrow \quad \Delta \tilde{R}_{i,k+1} = \Delta \tilde{R}_{ik}\,\mathrm{Exp}\big((\tilde{\boldsymbol{\omega}}_k - \boldsymbol{\eta}_g)\Delta t\big)$$

오차 정의: $\Delta \tilde{R}_{ik} = \Delta R_{ik}\,\mathrm{Exp}(\delta\phi_k)$ (left perturbation 사용 시):

$$\Delta \tilde{R}_{i,k+1} = \Delta R_{ik}\,\mathrm{Exp}(\delta\phi_k)\,\mathrm{Exp}\big((\tilde{\boldsymbol{\omega}}_k - \boldsymbol{\eta}_g)\Delta t\big)$$

BCH 1차 근사: $\mathrm{Exp}(\boldsymbol{a})\mathrm{Exp}(\boldsymbol{b}) \approx \mathrm{Exp}(\boldsymbol{a} + J_l^{-1}(\boldsymbol{a})\,\boldsymbol{b})$
(right perturbation 기준으로 J_r 등장)

결과

$$\delta\phi_{k+1} \approx \mathrm{Exp}(-\tilde{\boldsymbol{\omega}}_k \Delta t)\,\delta\phi_k - J_r^{-1}(\tilde{\boldsymbol{\omega}}_k \Delta t)\,\Delta t\,\boldsymbol{\eta}_g$$

$$\Rightarrow A_k[0:3, 0:3] = \mathrm{Exp}(-\tilde{\boldsymbol{\omega}}_k \Delta t),\ B_k[0:3, 0:3] = -J_r^{-1}(\tilde{\boldsymbol{\omega}}_k \Delta t)\,\Delta t$$

BCH 공식과 회전 오차 전파 — 상세 유도

목표: 다음을 선형화

$$\Delta R_{i,k+1} = \underbrace{\Delta R_{ik}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t)}_{\text{명목}} \quad \leftrightarrow \quad \Delta \tilde{R}_{i,k+1} = \Delta \tilde{R}_{ik}\,\mathrm{Exp}\big((\tilde{\boldsymbol{\omega}}_k - \boldsymbol{\eta}_g)\Delta t\big)$$

오차 정의: $\Delta \tilde{R}_{ik} = \Delta R_{ik}\,\mathrm{Exp}(\delta\phi_k)$ (left perturbation 사용 시):

$$\Delta \tilde{R}_{i,k+1} = \Delta R_{ik}\,\mathrm{Exp}(\delta\phi_k)\,\mathrm{Exp}\big((\tilde{\boldsymbol{\omega}}_k - \boldsymbol{\eta}_g)\Delta t\big)$$

BCH 1차 근사: $\mathrm{Exp}(\boldsymbol{a})\mathrm{Exp}(\boldsymbol{b}) \approx \mathrm{Exp}(\boldsymbol{a} + J_l^{-1}(\boldsymbol{a})\,\boldsymbol{b})$
(right perturbation 기준으로 J_r 등장)

결과

$$\delta\phi_{k+1} \approx \mathrm{Exp}(-\tilde{\boldsymbol{\omega}}_k \Delta t)\,\delta\phi_k - J_r^{-1}(\tilde{\boldsymbol{\omega}}_k \Delta t)\,\Delta t\,\boldsymbol{\eta}_g$$

$$\Rightarrow A_k[0:3, 0:3] = \mathrm{Exp}(-\tilde{\boldsymbol{\omega}}_k \Delta t), \;\; B_k[0:3, 0:3] = -J_r^{-1}(\tilde{\boldsymbol{\omega}}_k \Delta t)\,\Delta t$$

부호 주의
$\boldsymbol{\eta}_g$가 $\tilde{\boldsymbol{\omega}}_k$를 줄이는 방향이므로 $-J_r^{-1}$ (음의 부호). 공분산 전파에서는 $B_k Q_d B_k^\top$이므로 부호가 자동으로 제거됨.

속도 오차 전파 상세 유도

명목 속도 전파:

$$\Delta v_{i,k+1} = \Delta v_{ik} + \Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t$$

실제 속도 전파 (노이즈 포함):

$$\Delta \tilde{v}_{i,k+1} = \Delta \tilde{v}_{ik} + \underbrace{\Delta \tilde{R}_{ik}}_{\approx \Delta R_{ik}(I + [\delta\phi_k]_\times)} (\tilde{\boldsymbol{a}}_k - \boldsymbol{\eta}_a)\,\Delta t$$

속도 오차 전파 상세 유도

명목 속도 전파:

$$\Delta v_{i,k+1} = \Delta v_{ik} + \Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t$$

실제 속도 전파 (노이즈 포함):

$$\Delta \tilde{v}_{i,k+1} = \Delta \tilde{v}_{ik} + \underbrace{\Delta \tilde{R}_{ik}}_{\approx \Delta R_{ik}(I + [\delta\phi_k]_\times)} (\tilde{\boldsymbol{a}}_k - \boldsymbol{\eta}_a)\,\Delta t$$

오차 $\delta v_k = \Delta \tilde{v}_{ik} - \Delta v_{ik}$에 대해 1차 근사:

$$\begin{aligned}\delta v_{k+1} &= \delta v_k + \Delta R_{ik}\,[\delta\phi_k]_\times\,\tilde{\boldsymbol{a}}_k\,\Delta t - \Delta R_{ik}\,\boldsymbol{\eta}_a\,\Delta t \\ &= \delta v_k - \Delta R_{ik}\,[\tilde{\boldsymbol{a}}_k]_\times\,\delta\phi_k\,\Delta t - \Delta R_{ik}\,\boldsymbol{\eta}_a\,\Delta t\end{aligned}$$

($[\delta\phi]_\times\,\boldsymbol{a} = -[\boldsymbol{a}]_\times\,\delta\phi$ 성질 사용)

속도 오차 전파 상세 유도

명목 속도 전파:

$$\Delta v_{i,k+1} = \Delta v_{ik} + \Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t$$

실제 속도 전파 (노이즈 포함):

$$\Delta \tilde{v}_{i,k+1} = \Delta \tilde{v}_{ik} + \underbrace{\Delta \tilde{R}_{ik}}_{\approx \Delta R_{ik}(I + [\delta\phi_k]_\times)} (\tilde{\boldsymbol{a}}_k - \boldsymbol{\eta}_a)\,\Delta t$$

오차 $\delta v_k = \Delta \tilde{v}_{ik} - \Delta v_{ik}$에 대해 1차 근사:

$$\begin{aligned}\delta v_{k+1} &= \delta v_k + \Delta R_{ik}\,[\delta\phi_k]_\times\,\tilde{\boldsymbol{a}}_k\,\Delta t - \Delta R_{ik}\,\boldsymbol{\eta}_a\,\Delta t \\ &= \delta v_k - \Delta R_{ik}\,[\tilde{\boldsymbol{a}}_k]_\times\,\delta\phi_k\,\Delta t - \Delta R_{ik}\,\boldsymbol{\eta}_a\,\Delta t\end{aligned}$$

($[\delta\phi]_\times\,\boldsymbol{a} = -[\boldsymbol{a}]_\times\,\delta\phi$ 성질 사용)

A_k 속도 블록

$$A_k[3{:}6,\,0{:}3] = -\Delta R_{ik}\,[\tilde{\boldsymbol{a}}_k]_\times\,\Delta t, \qquad A_k[3{:}6,\,3{:}6] = I$$

B_k 속도 블록

$$B_k[3{:}6,\,3{:}6] = -\Delta R_{ik}\,\Delta t \quad \text{(accel noise 컬럼)}$$

위치 오차 전파 상세 유도

명목 위치 전파:

$$\Delta p_{i,k+1} = \Delta p_{ik} + \Delta v_{ik}\,\Delta t + \tfrac{1}{2}\,\Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t^2$$

위치 오차 전파 상세 유도

명목 위치 전파:

$$\Delta p_{i,k+1} = \Delta p_{ik} + \Delta v_{ik}\,\Delta t + \tfrac{1}{2}\,\Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t^2$$

오차 $\delta p_k = \Delta\tilde{p}_{ik} - \Delta p_{ik}$:

$$\begin{aligned}\delta p_{k+1} &= \delta p_k + \delta v_k\,\Delta t + \tfrac{1}{2}\,\Delta R_{ik}\,\left[\delta\phi_k\right]_\times\,\tilde{\boldsymbol{a}}_k\,\Delta t^2 - \tfrac{1}{2}\,\Delta R_{ik}\,\boldsymbol{\eta}_a\,\Delta t^2 \\ &= \delta p_k + \delta v_k\,\Delta t - \tfrac{1}{2}\,\Delta R_{ik}\,\left[\tilde{\boldsymbol{a}}_k\right]_\times\,\delta\phi_k\,\Delta t^2 - \tfrac{1}{2}\,\Delta R_{ik}\,\boldsymbol{\eta}_a\,\Delta t^2\end{aligned}$$

위치 오차 전파 상세 유도

명목 위치 전파:

$$\Delta p_{i,k+1} = \Delta p_{ik} + \Delta v_{ik}\,\Delta t + \tfrac{1}{2}\,\Delta R_{ik}\,\tilde{\boldsymbol{a}}_k\,\Delta t^2$$

오차 $\delta p_k = \Delta\tilde{p}_{ik} - \Delta p_{ik}$:

$$\begin{aligned}\delta p_{k+1} &= \delta p_k + \delta v_k\,\Delta t + \tfrac{1}{2}\,\Delta R_{ik}\,\left[\delta\phi_k\right]_\times\,\tilde{\boldsymbol{a}}_k\,\Delta t^2 - \tfrac{1}{2}\,\Delta R_{ik}\,\boldsymbol{\eta}_a\,\Delta t^2 \\ &= \delta p_k + \delta v_k\,\Delta t - \tfrac{1}{2}\,\Delta R_{ik}\,\left[\tilde{\boldsymbol{a}}_k\right]_\times\,\delta\phi_k\,\Delta t^2 - \tfrac{1}{2}\,\Delta R_{ik}\,\boldsymbol{\eta}_a\,\Delta t^2\end{aligned}$$

A_k **위치 블록**

$$\begin{aligned}A_k[6{:}9,\,0{:}3] &= -\tfrac{1}{2}\,\Delta R_{ik}\,\left[\tilde{\boldsymbol{a}}_k\right]_\times\,\Delta t^2 \\ A_k[6{:}9,\,3{:}6] &= I\,\Delta t \\ A_k[6{:}9,\,6{:}9] &= I\end{aligned}$$

B_k **위치 블록**

$$B_k[6{:}9,\,3{:}6] = -\tfrac{1}{2}\,\Delta R_{ik}\,\Delta t^2 \quad \text{(accel noise 컬럼)}$$

체인룰 출발:

$$\Delta R_{i,k+1}(b_g) = \Delta R_{ik}(b_g)\ \mathrm{Exp}\big((\boldsymbol{\omega}_{m,k} - b_g)\Delta t\big)$$

$J_{b_g}^{\Delta R}$ 재귀식 상세 유도

체인룰 출발:

$$\Delta R_{i,k+1}(b_g) = \Delta R_{ik}(b_g)\ \mathrm{Exp}\big((\boldsymbol{\omega}_{m,k} - b_g)\Delta t\big)$$

b_g에 대한 편미분 (right perturbation $\epsilon \to 0$ 극한):

$$\frac{\partial}{\partial b_g}\left[\Delta R_{ik}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t)\right]$$

체인룰:

$$= \underbrace{\frac{\partial\left[\Delta R_{ik}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k\Delta t)\right]}{\partial\left[\Delta R_{ik}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k\Delta t)\right]}}_{\text{SO(3) 결합 미분}} \cdot \left(\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k\Delta t)^{\top}\frac{\partial \Delta R_{ik}}{\partial b_g} + J_r^{-1}(\tilde{\boldsymbol{\omega}}_k\Delta t)\cdot(-\Delta t)\right)$$

$J_{b_g}^{\Delta R}$ 재귀식 상세 유도

체인룰 출발:

$$\Delta R_{i,k+1}(b_g) = \Delta R_{ik}(b_g)\ \mathrm{Exp}\big((\boldsymbol{\omega}_{m,k} - b_g)\Delta t\big)$$

b_g에 대한 편미분 (right perturbation $\epsilon \to 0$ 극한):

$$\frac{\partial}{\partial b_g}\left[\Delta R_{ik}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k \Delta t)\right]$$

체인룰:

$$= \underbrace{\frac{\partial\left[\Delta R_{ik}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k\Delta t)\right]}{\partial\left[\Delta R_{ik}\,\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k\Delta t)\right]}}_{\mathrm{SO(3)\ 결합\ 미분}} \cdot \left(\mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k\Delta t)^{\top}\frac{\partial \Delta R_{ik}}{\partial b_g} + J_r^{-1}(\tilde{\boldsymbol{\omega}}_k\Delta t)\cdot(-\Delta t)\right)$$

최종 재귀식

$$\boxed{J_{b_g}^{\Delta R_{k+1}} = \mathrm{Exp}(-\tilde{\boldsymbol{\omega}}_k\Delta t)\ J_{b_g}^{\Delta R_k} - J_r^{-1}(\tilde{\boldsymbol{\omega}}_k\Delta t)\,\Delta t}$$

- 첫 항: 이전 Jacobian을 현재 frame으로 회전 변환
- 둘째 항: 현재 스텝에서 b_g 변화가 $\tilde{\boldsymbol{\omega}}$에 미치는 직접 효과

6개 Jacobian 재귀식 일람표

Jacobian	재귀식	초기값
$J_{b_g}^{\Delta R}$	$\mathrm{Exp}(-\tilde{\boldsymbol{\omega}}\Delta t)\, J^R - J_r^{-1}\Delta t$	$0_{3\times3}$
$J_{b_g}^{\Delta v}$	$J^v - \Delta R\, [\tilde{\boldsymbol{a}}]_\times\, J_{b_g}^{\Delta R}\, \Delta t$	$0_{3\times3}$
$J_{b_a}^{\Delta v}$	$J_a^v - \Delta R\, \Delta t$	$0_{3\times3}$
$J_{b_g}^{\Delta p}$	$J^p + J_{b_g}^{\Delta v}\, \Delta t - \frac{1}{2}\Delta R\, [\tilde{\boldsymbol{a}}]_\times\, J_{b_g}^{\Delta R}\, \Delta t^2$	$0_{3\times3}$
$J_{b_a}^{\Delta p}$	$J_a^p + J_{b_a}^{\Delta v}\, \Delta t - \frac{1}{2}\Delta R\, \Delta t^2$	$0_{3\times3}$
$J_{b_a}^{\Delta R}$	$= 0$	$0_{3\times3}$

6개 Jacobian 재귀식 일람표

Jacobian	재귀식	초기값
$J_{b_g}^{\Delta R}$	$\mathrm{Exp}(-\tilde{\boldsymbol{\omega}}\Delta t)\, J^R - J_r^{-1}\Delta t$	$0_{3\times3}$
$J_{b_g}^{\Delta v}$	$J^v - \Delta R\, [\tilde{\boldsymbol{a}}]_\times\, J_{b_g}^{\Delta R}\, \Delta t$	$0_{3\times3}$
$J_{b_a}^{\Delta v}$	$J_a^v - \Delta R\, \Delta t$	$0_{3\times3}$
$J_{b_g}^{\Delta p}$	$J^p + J_{b_g}^{\Delta v}\, \Delta t - \frac{1}{2}\Delta R\, [\tilde{\boldsymbol{a}}]_\times\, J_{b_g}^{\Delta R}\, \Delta t^2$	$0_{3\times3}$
$J_{b_a}^{\Delta p}$	$J_a^p + J_{b_a}^{\Delta v}\, \Delta t - \frac{1}{2}\Delta R\, \Delta t^2$	$0_{3\times3}$
$J_{b_a}^{\Delta R}$	$= 0$	$0_{3\times3}$

계산 순서 (의존성 주의)

$$J_{b_g}^{\Delta R_k} \longrightarrow J_{b_g}^{\Delta v_k} \longrightarrow J_{b_g}^{\Delta p_k}$$

각 스텝에서 이 순서를 지켜야 함.

C++ 구현 예시 — 6개 Jacobian 재귀 루프

Listing 4: Preintegration Jacobians 재귀 구현 (교육용 의사 코드)

```
// 초기화
Eigen::Matrix3d JR  = Eigen::Matrix3d::Zero();  // d(DeltaR)/d(bg)
Eigen::Matrix3d Jvg = Eigen::Matrix3d::Zero();  // d(Deltav)/d(bg)
Eigen::Matrix3d Jva = Eigen::Matrix3d::Zero();  // d(Deltav)/d(ba)
Eigen::Matrix3d Jpg = Eigen::Matrix3d::Zero();  // d(Deltap)/d(bg)
Eigen::Matrix3d Jpa = Eigen::Matrix3d::Zero();  // d(Deltap)/d(ba)

Eigen::Matrix3d dR_accum = Eigen::Matrix3d::Identity(); // Delta R_ik

for (const auto& meas : imu_measurements) {
    double dt     = meas.dt;
    Eigen::Vector3d omegatil = meas.gyro - bg;
    Eigen::Vector3d atil     = meas.accel - ba;
    Eigen::Matrix3d dR = Exp(omegatil * dt);           // Exp(~omega * dt)
    Eigen::Matrix3d Jr_inv = rightJacobianInverse(omegatil * dt);

    // 위치속도/ Jacobian 순서( 중요: 먼저저장후업데이트   )
    Jpg = Jpg + Jvg * dt
              - 0.5 * dR_accum * skew(atil) * JR * dt * dt;
    Jpa = Jpa + Jva * dt - 0.5 * dR_accum * dt * dt;
    Jvg = Jvg - dR_accum * skew(atil) * JR * dt;
    Jva = Jva - dR_accum * dt;

    // 회전 Jacobian 마지막에( 업데이트)
    JR  = dR.transpose() * JR - Jr_inv * dt;

    dR_accum = dR_accum * dR;  // Delta R_{i,k+1}
}
// 결과: JR, Jvg, Jva, Jpg, Jpa 사용가능
```

Preintegration과 ESKF의 관계 — 비교

ESKF Predict

- **전역 상태** (R, p, v, b_g, b_a) 추적
- 매 IMU 스텝마다 $\boldsymbol{P}$ 전파
- 측정 업데이트: GPS, camera 등
- Filter 구조: 단일 epoch
- 재선형화: 없음 (매 스텝 새로 선형화)

Preintegration (Factor Graph)

- **상대 측정** $(\Delta R, \Delta v, \Delta p)$ 축적
- 키프레임 간 Σ_{ij} 전파
- 스무딩: 과거 상태도 재추정
- Factor 구조: 다수 epoch 동시
- 재선형화: 반복 최적화에서 가능

Preintegration과 ESKF의 관계 — 비교

ESKF Predict

- **전역 상태** (R, p, v, b_g, b_a) 추적
- 매 IMU 스텝마다 $\boldsymbol{P}$ 전파
- 측정 업데이트: GPS, camera 등
- Filter 구조: 단일 epoch
- 재선형화: 없음 (매 스텝 새로 선형화)

Preintegration (Factor Graph)

- **상대 측정** $(\Delta R, \Delta v, \Delta p)$ 축적
- 키프레임 간 Σ_{ij} 전파
- 스무딩: 과거 상태도 재추정
- Factor 구조: 다수 epoch 동시
- 재선형화: 반복 최적화에서 가능

둘의 공통점

- 동일한 IMU noise 모델 $(\sigma_g, \sigma_a, \sigma_{bg}, \sigma_{ba})$
- A_k, B_k 행렬 구조가 유사 (ESKF의 $\boldsymbol{F}, \boldsymbol{G}$에 대응)
- 바이어스 상태가 핵심 불확실도 요소

수치 예제 — EuRoC V1_01 파라미터로 계산

사용 파라미터 (ASL Leica MoCap 방 환경, IMU: ADIS16448):

$$\sigma_g = 1.6968 \times 10^{-4} \text{ rad/s}/\sqrt{\text{Hz}}$$
$$\sigma_a = 2.0000 \times 10^{-3} \text{ m/s}^2/\sqrt{\text{Hz}}$$
$$\sigma_{bg} = 1.9393 \times 10^{-5} \text{ rad/s}^2/\sqrt{\text{Hz}}$$
$$\sigma_{ba} = 3.0000 \times 10^{-3} \text{ m/s}^3/\sqrt{\text{Hz}}$$
$$f_{IMU} = 200 \text{ Hz}, \quad \Delta t = 0.005 \text{ s}$$

수치 예제 — EuRoC V1_01 파라미터로 계산

사용 파라미터 (ASL Leica MoCap 방 환경, IMU: ADIS16448):

$$\sigma_g = 1.6968 \times 10^{-4} \text{ rad/s}/\sqrt{\text{Hz}}$$
$$\sigma_a = 2.0000 \times 10^{-3} \text{ m/s}^2/\sqrt{\text{Hz}}$$
$$\sigma_{bg} = 1.9393 \times 10^{-5} \text{ rad/s}^2/\sqrt{\text{Hz}}$$
$$\sigma_{ba} = 3.0000 \times 10^{-3} \text{ m/s}^3/\sqrt{\text{Hz}}$$
$$f_{IMU} = 200 \text{ Hz}, \quad \Delta t = 0.005 \text{ s}$$

이산 노이즈 행렬 대각 원소

$$Q_d[0:3, 0:3] = \frac{(1.6968 \times 10^{-4})^2}{0.005} I = 5.77 \times 10^{-6} I \text{ rad}^2$$

$$Q_d[3:6, 3:6] = \frac{(2.0 \times 10^{-3})^2}{0.005} I = 8.0 \times 10^{-4} I \text{ (m/s)}^2$$

수치 예제 — EuRoC V1_01 파라미터로 계산

사용 파라미터 (ASL Leica MoCap 방 환경, IMU: ADIS16448):

$$\sigma_g = 1.6968 \times 10^{-4} \text{ rad/s}/\sqrt{\text{Hz}}$$
$$\sigma_a = 2.0000 \times 10^{-3} \text{ m/s}^2/\sqrt{\text{Hz}}$$
$$\sigma_{bg} = 1.9393 \times 10^{-5} \text{ rad/s}^2/\sqrt{\text{Hz}}$$
$$\sigma_{ba} = 3.0000 \times 10^{-3} \text{ m/s}^3/\sqrt{\text{Hz}}$$
$$f_{IMU} = 200 \text{ Hz}, \quad \Delta t = 0.005 \text{ s}$$

이산 노이즈 행렬 대각 원소

$$Q_d[0:3, 0:3] = \frac{(1.6968 \times 10^{-4})^2}{0.005} I = 5.77 \times 10^{-6} I \text{ rad}^2$$
$$Q_d[3:6, 3:6] = \frac{(2.0 \times 10^{-3})^2}{0.005} I = 8.0 \times 10^{-4} I \text{ (m/s)}^2$$

0.5초 preintegration (100 스텝) 후 Σ_{ij} 대각 크기 (근사)

$$\sqrt{\Sigma_{\phi\phi}} \approx 0.02 \text{ deg}, \quad \sqrt{\Sigma_{vv}} \approx 0.01 \text{ m/s}, \quad \sqrt{\Sigma_{pp}} \approx 0.002 \text{ m}$$

(실제 값은 운동 궤적과 ΔR_{ik}에 따라 달라짐)

바이어스 업데이트 후 선형 보정 — 수치 예제

상황: ESKF camera update 후 바이어스 추정 변화

$$\delta b_g = [0.001, -0.002, 0.0005]^\top \text{ rad/s}, \quad \delta b_a = [0.01, 0.005, -0.008]^\top \text{ m/s}^2$$

바이어스 업데이트 후 선형 보정 — 수치 예제

상황: ESKF camera update 후 바이어스 추정 변화

$$\delta b_g = [0.001, -0.002, 0.0005]^\top \text{ rad/s}, \quad \delta b_a = [0.01, 0.005, -0.008]^\top \text{ m/s}^2$$

1초 preintegration에서 보정 크기 추정

$$\|J_{b_g}^{\Delta R}\,\delta b_g\| \approx \|\delta b_g\| \cdot 1\text{ s} = 0.00224 \text{ rad} \approx 0.13 \text{ deg}$$

$$\|J_{b_a}^{\Delta v}\,\delta b_a\| \approx \|\delta b_a\| \cdot 1\text{ s} = 0.0137 \text{ m/s}$$

$$\|J_{b_a}^{\Delta p}\,\delta b_a\| \approx \tfrac{1}{2}\|\delta b_a\| \cdot 1^2\text{ s}^2 = 0.007 \text{ m}$$

바이어스 업데이트 후 선형 보정 — 수치 예제

상황: ESKF camera update 후 바이어스 추정 변화

$$\delta b_g = [0.001, -0.002, 0.0005]^\top \text{ rad/s}, \quad \delta b_a = [0.01, 0.005, -0.008]^\top \text{ m/s}^2$$

1초 preintegration에서 보정 크기 추정

$$\|J_{b_g}^{\Delta R}\,\delta b_g\| \approx \|\delta b_g\| \cdot 1\text{ s} = 0.00224 \text{ rad} \approx 0.13 \text{ deg}$$

$$\|J_{b_a}^{\Delta v}\,\delta b_a\| \approx \|\delta b_a\| \cdot 1\text{ s} = 0.0137 \text{ m/s}$$

$$\|J_{b_a}^{\Delta p}\,\delta b_a\| \approx \tfrac{1}{2}\|\delta b_a\| \cdot 1^2\text{ s}^2 = 0.007 \text{ m}$$

판단

0.13 deg 회전 보정은 선형 근사 범위 내.
0.007 m 위치 보정 $\approx$ GPS 정밀도 수준 $\Rightarrow$ 선형 보정으로 충분.
만약 $\|\delta b_g\| > 0.1$ rad/s 이상이면 재적분 권장.

Factor Residual Jacobian — 바이어스 열 상세

잔차 $\boldsymbol{r}_{ij} \in \mathbb{R}^9$의 바이어스에 대한 Jacobian:

$\partial r_{\Delta R} / \partial \delta b_g$ **(3×3)**

$$\frac{\partial r_{\Delta R}}{\partial \delta b_g} = -J_r^{-1}(\mathrm{Log}(\Delta \hat{R}_{ij}^\top R_i^\top R_j))\ J_{b_g}^{\Delta R_{ij}}$$

($\Delta \hat{R}$가 b_g에 의존하므로 체인룰)

Factor Residual Jacobian — 바이어스 열 상세

잔차 $\boldsymbol{r}_{ij} \in \mathbb{R}^9$의 바이어스에 대한 Jacobian:

$\partial r_{\Delta R}/\partial \delta b_g$ **(3×3)**

$$\frac{\partial r_{\Delta R}}{\partial \delta b_g} = -J_r^{-1}(\mathrm{Log}(\Delta \hat{R}_{ij}^\top R_i^\top R_j))\; J_{b_g}^{\Delta R_{ij}}$$

($\Delta \hat{R}$가 b_g에 의존하므로 체인룰)

$\partial r_{\Delta v}/\partial \delta b_g$**,** $\partial r_{\Delta v}/\partial \delta b_a$ **(3×3 각)**

$$\frac{\partial r_{\Delta v}}{\partial \delta b_g} = -J_{b_g}^{\Delta v_{ij}}, \qquad \frac{\partial r_{\Delta v}}{\partial \delta b_a} = -J_{b_a}^{\Delta v_{ij}}$$

Factor Residual Jacobian — 바이어스 열 상세

잔차 $\boldsymbol{r}_{ij} \in \mathbb{R}^9$의 바이어스에 대한 Jacobian:

$\partial r_{\Delta R}/\partial \delta b_g$ **(3×3)**

$$\frac{\partial r_{\Delta R}}{\partial \delta b_g} = -J_r^{-1}(\mathrm{Log}(\Delta \hat{R}_{ij}^\top R_i^\top R_j))\ J_{b_g}^{\Delta R_{ij}}$$

($\Delta \hat{R}$가 b_g에 의존하므로 체인룰)

$\partial r_{\Delta v}/\partial \delta b_g$**,** $\partial r_{\Delta v}/\partial \delta b_a$ **(3×3 각)**

$$\frac{\partial r_{\Delta v}}{\partial \delta b_g} = -J_{b_g}^{\Delta v_{ij}}, \qquad \frac{\partial r_{\Delta v}}{\partial \delta b_a} = -J_{b_a}^{\Delta v_{ij}}$$

$\partial r_{\Delta p}/\partial \delta b_g$**,** $\partial r_{\Delta p}/\partial \delta b_a$ **(3×3 각)**

$$\frac{\partial r_{\Delta p}}{\partial \delta b_g} = -J_{b_g}^{\Delta p_{ij}}, \qquad \frac{\partial r_{\Delta p}}{\partial \delta b_a} = -J_{b_a}^{\Delta p_{ij}}$$

Factor Residual Jacobian — 바이어스 열 상세

잔차 $\boldsymbol{r}_{ij} \in \mathbb{R}^9$의 바이어스에 대한 Jacobian:

$\partial r_{\Delta R}/\partial \delta b_g$ **(3×3)**

$$\frac{\partial r_{\Delta R}}{\partial \delta b_g} = -J_r^{-1}(\mathrm{Log}(\Delta\hat{R}_{ij}^\top R_i^\top R_j))\; J_{b_g}^{\Delta R_{ij}}$$

($\Delta\hat{R}$가 b_g에 의존하므로 체인룰)

$\partial r_{\Delta v}/\partial \delta b_g$**,** $\partial r_{\Delta v}/\partial \delta b_a$ **(3×3 각)**

$$\frac{\partial r_{\Delta v}}{\partial \delta b_g} = -J_{b_g}^{\Delta v_{ij}}, \qquad \frac{\partial r_{\Delta v}}{\partial \delta b_a} = -J_{b_a}^{\Delta v_{ij}}$$

$\partial r_{\Delta p}/\partial \delta b_g$**,** $\partial r_{\Delta p}/\partial \delta b_a$ **(3×3 각)**

$$\frac{\partial r_{\Delta p}}{\partial \delta b_g} = -J_{b_g}^{\Delta p_{ij}}, \qquad \frac{\partial r_{\Delta p}}{\partial \delta b_a} = -J_{b_a}^{\Delta p_{ij}}$$

전체 bias Jacobian 블록 $\in \mathbb{R}^{9\times 6}$

$$\frac{\partial \boldsymbol{r}_{ij}}{\partial [\delta {b_g}^\top, \delta {b_a}^\top]^\top} = \begin{pmatrix} -J_r^{-1}(\cdots) J_{b_g}^{\Delta R} & 0 \\ -J_{b_g}^{\Delta v} & -J_{b_a}^{\Delta v} \\ -J_{b_g}^{\Delta p} & -J_{b_a}^{\Delta p} \end{pmatrix}$$

Covariance Propagation vs ESKF P — 어디서 차이가 나는가

ESKF $P \in \mathbb{R}^{15\times 15}$ (pose + bias)

$$P = \begin{pmatrix} P_{\xi\xi} & P_{\xi b} \\ P_{b\xi} & P_{bb} \end{pmatrix}$$

상태: $[\delta\phi^\top, \delta v^\top, \delta p^\top, \delta b_g^\top, \delta b_a^\top]^\top \in \mathbb{R}^{15}$

Covariance Propagation vs ESKF P — 어디서 차이가 나는가

ESKF $P \in \mathbb{R}^{15\times15}$ (pose + bias)

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{P}_{\xi\xi} & \boldsymbol{P}_{\xi b} \\ \boldsymbol{P}_{b\xi} & \boldsymbol{P}_{bb} \end{pmatrix}$$

상태: $[\delta\phi^\top, \delta v^\top, \delta p^\top, \delta b_g^\top, \delta b_a^\top]^\top \in \mathbb{R}^{15}$

Preintegration $\Sigma_{ij} \in \mathbb{R}^{9\times9}$ (pose만)

$$\Sigma_{ij} = \begin{pmatrix} \Sigma_{\phi\phi} & \Sigma_{\phi v} & \Sigma_{\phi p} \\ \Sigma_{v\phi} & \Sigma_{vv} & \Sigma_{vp} \\ \Sigma_{p\phi} & \Sigma_{pv} & \Sigma_{pp} \end{pmatrix}$$

바이어스 불확실도는 **별도 추적**: $\Sigma_b = \text{diag}(\sigma_{bg}^2 \Delta t_{ij}, \sigma_{ba}^2 \Delta t_{ij}) I$

Covariance Propagation vs ESKF P — 어디서 차이가 나는가

ESKF $\boldsymbol{P} \in \mathbb{R}^{15\times15}$ (pose + bias)

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{P}_{\xi\xi} & \boldsymbol{P}_{\xi b} \\ \boldsymbol{P}_{b\xi} & \boldsymbol{P}_{bb} \end{pmatrix}$$

상태: $[\delta\phi^\top, \delta v^\top, \delta p^\top, \delta b_g^\top, \delta b_a^\top]^\top \in \mathbb{R}^{15}$

Preintegration $\Sigma_{ij} \in \mathbb{R}^{9\times9}$ (pose만)

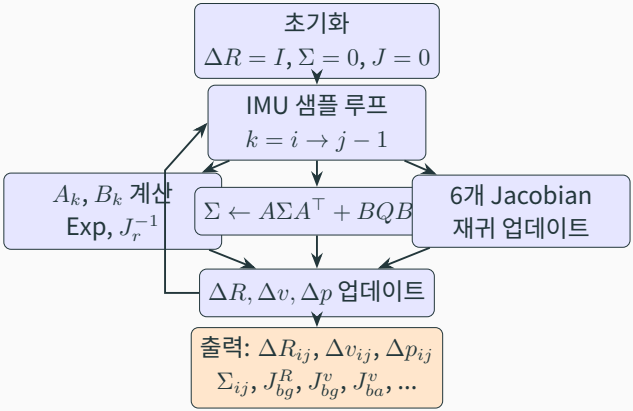
$$\Sigma_{ij} = \begin{pmatrix} \Sigma_{\phi\phi} & \Sigma_{\phi v} & \Sigma_{\phi p} \\ \Sigma_{v\phi} & \Sigma_{vv} & \Sigma_{vp} \\ \Sigma_{p\phi} & \Sigma_{pv} & \Sigma_{pp} \end{pmatrix}$$

바이어스 불확실도는 **별도 추적**: $\Sigma_b = \text{diag}(\sigma_{bg}^2 \Delta t_{ij}, \sigma_{ba}^2 \Delta t_{ij}) I$

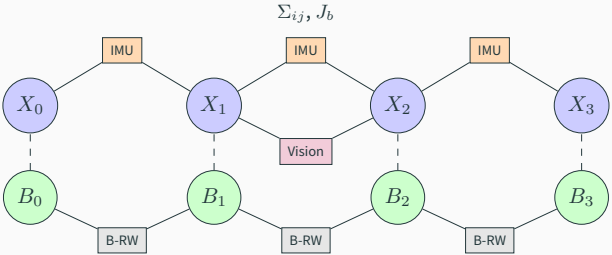
핵심 차이

- ESKF: 바이어스 오차가 $\boldsymbol{P}_{\xi b}$ 교차항을 통해 위치/속도 공분산과 결합
- Preintegration: 바이어스 효과를 **Jacobian으로 분리**, 공분산은 pose 부분만
- Factor graph에서 바이어스 공분산은 bias random walk factor가 별도로 제공

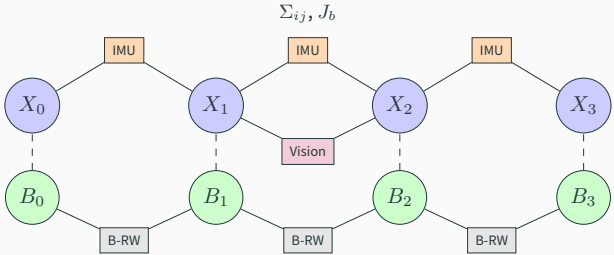
전체 알고리즘 흐름도 — 코드 관점



Factor Graph 전체 구조에서 Preintegration Factor의 위치



Factor Graph 전체 구조에서 Preintegration Factor의 위치



- IMU factor: Σ_{ij}, 6개 Jacobian 내장
- Bias RW factor: $\Sigma_b = \sigma_{bg}^2 \Delta t_{ij}\, I$ 제공
- Dashed: 바이어스 B_i가 IMU factor 잔차에 영향 (Jacobian을 통해)
- Vision factor: 회전/위치 구속 (별도 Σ_{vis})

실습 과제 안내

과제 1: Jacobian 재귀 구현
Python 또는 C++ 로 IMU 100 스텝에 대해:

1. $\Delta R_{ij}, \Delta v_{ij}, \Delta p_{ij}$ 계산
2. 6개 Bias Jacobian 동시 계산
3. Σ_{ij} 전파

제공 파일: `preintegration_skeleton.py`

실습 과제 안내

과제 1: Jacobian 재귀 구현
Python 또는 C++ 로 IMU 100 스텝에 대해:

1. $\Delta R_{ij}, \Delta v_{ij}, \Delta p_{ij}$ 계산
2. 6개 Bias Jacobian 동시 계산
3. Σ_{ij} 전파

제공 파일: `preintegration_skeleton.py`

과제 2: 선형 보정 검증

1. $\delta b_g = [0.01, 0, 0]^\top$ rad/s 적용
2. 선형 보정 vs 실제 재적분 결과 비교
3. 오차가 2차($\propto \|\delta b\|^2$)임을 수치로 확인

실습 과제 안내

과제 1: Jacobian 재귀 구현
Python 또는 C++ 로 IMU 100 스텝에 대해:

1. $\Delta R_{ij}, \Delta v_{ij}, \Delta p_{ij}$ 계산
2. 6개 Bias Jacobian 동시 계산
3. Σ_{ij} 전파

제공 파일: `preintegration_skeleton.py`

과제 2: 선형 보정 검증

1. $\delta b_g = [0.01, 0, 0]^\top$ rad/s 적용
2. 선형 보정 vs 실제 재적분 결과 비교
3. 오차가 2차($\propto \|\delta b\|^2$)임을 수치로 확인

선택 과제: GTSAM 연동
GTSAM `PreintegratedImuMeasurements` 사용하여
factor graph 구성 후 `iSAM2`로 최적화.
EuRoC V1_01 데이터셋으로 ATE 측정.

핵심 참고문헌

1. **Forster et al. (2017)**
   “On-Manifold Preintegration for Real-Time Visual–Inertial Odometry”
   IEEE Transactions on Robotics, 33(1):1–21.
   이 강의의 주요 수식 출처 — eq. (33), (35)–(37), (44)–(46)

2. **Lupton & Sukkarieh (2012)**
   “Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions”
   IEEE TRO, 28(1):61–76.
   (비교 대상: 유클리드 preintegration)

3. **Geneva et al. (2020)**
   “OpenVINS: A Research Platform for Visual-Inertial Estimation”
   ICRA 2020.
   `github.com/rpng/open_vins` (GPLv3)

4. **Sola, Deray, Atchuthan (2018)**
   “A micro Lie theory for state estimation in robotics”
   arXiv:1812.01537 — BCH, J_r, J_r^{-1} 유도 참고

6회차: 멀티센서 Update

지난 시간 한 줄

ESKF Predict = 명목 상태(비선형) + 공분산(선형 $\boldsymbol{F}_d \boldsymbol{P} \boldsymbol{F}_d^\top + \boldsymbol{Q}_d$).
IMU는 잡음+바이어스가 섞인 신호 — $\boldsymbol{F}$로 오차 동역학을 선형화.

오늘: **센서 Update — z가 들어오면 $\delta\hat{x}$를 어떻게 구하는가**

- GPS (위치, 속도), 휠 인코더 (속도), 카메라 (피처 잔차)
- 센서마다 **H 행렬**이 다름 — 측정을 15D 오차 상태로 이어주는 다리
- Mahalanobis 기반 이상값 제거

일반 Update 공식

ESKF Update step의 구조:

$$
\begin{aligned}
\boldsymbol{r} &= \boldsymbol{z} - h(\hat{\boldsymbol{x}}) \quad \text{(residual: 측정값 } - \text{ 예측값)} \\
\boldsymbol{S} &= \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^{\top} + \boldsymbol{R} \\
\boldsymbol{K} &= \boldsymbol{P}\boldsymbol{H}^{\top}\boldsymbol{S}^{-1} \\
\delta\hat{\boldsymbol{x}} &= \boldsymbol{K}\boldsymbol{r} \\
\boldsymbol{P}^{+} &= (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})^{\top} + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^{\top}
\end{aligned}
$$

일반 Update 공식

ESKF Update step의 구조:

$$
\begin{aligned}
\boldsymbol{r} &= \boldsymbol{z} - h(\hat{\boldsymbol{x}}) \quad \text{(residual: 측정값 } - \text{ 예측값)} \\
\boldsymbol{S} &= \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^\top + \boldsymbol{R} \\
\boldsymbol{K} &= \boldsymbol{P}\boldsymbol{H}^\top \boldsymbol{S}^{-1} \\
\delta\hat{\boldsymbol{x}} &= \boldsymbol{K}\boldsymbol{r} \\
\boldsymbol{P}^+ &= (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})^\top + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^\top
\end{aligned}
$$

센서마다 달라지는 것: $h(\hat{\boldsymbol{x}})$**와** $\boldsymbol{H}$. 나머지는 동일.

$\boldsymbol{H}$의 의미
$\boldsymbol{H} = \partial h / \partial \delta\boldsymbol{x}\big|_{\delta\boldsymbol{x}=0}$ — 15D 오차 상태가 측정 잔차에 얼마나 영향을 미치는가.

GPS 위치 측정 — 가장 단순한 H

측정 모델: $\boldsymbol{z}_{\text{gps}} = \boldsymbol{p}_{\text{true}} + \boldsymbol{v}_{\text{gps}}$

명목 예측: $h(\hat{\boldsymbol{x}}) = \hat{\boldsymbol{p}}$

잔차: $\boldsymbol{r} = \boldsymbol{z}_{\text{gps}} - \hat{\boldsymbol{p}} = (\hat{\boldsymbol{p}} + \delta\boldsymbol{p}) - \hat{\boldsymbol{p}} + \boldsymbol{v}_{\text{gps}} = \delta\boldsymbol{p} + \boldsymbol{v}_{\text{gps}}$

GPS 위치 측정 — 가장 단순한 H

측정 모델: $\boldsymbol{z}_{\text{gps}} = \boldsymbol{p}_{\text{true}} + \boldsymbol{v}_{\text{gps}}$

명목 예측: $h(\hat{\boldsymbol{x}}) = \hat{\boldsymbol{p}}$

잔차: $\boldsymbol{r} = \boldsymbol{z}_{\text{gps}} - \hat{\boldsymbol{p}} = (\hat{\boldsymbol{p}} + \delta\boldsymbol{p}) - \hat{\boldsymbol{p}} + \boldsymbol{v}_{\text{gps}} = \delta\boldsymbol{p} + \boldsymbol{v}_{\text{gps}}$

H 행렬 (3×15, 15D 상태 순서 $[\delta\boldsymbol{\phi},\, \delta\boldsymbol{p},\, \delta\boldsymbol{v},\, \delta\boldsymbol{b}_g,\, \delta\boldsymbol{b}_a]$):

$$\boldsymbol{H}_{\text{pos}} = \begin{pmatrix} \underbrace{0_{3\times3}}_{\delta\boldsymbol{\phi}} & \underbrace{\boldsymbol{I}_{3\times3}}_{\delta\boldsymbol{p}} & \underbrace{0_{3\times3}}_{\delta\boldsymbol{v}} & \underbrace{0_{3\times3}}_{\delta\boldsymbol{b}_g} & \underbrace{0_{3\times3}}_{\delta\boldsymbol{b}_a} \end{pmatrix}$$

GPS 위치 측정 — 가장 단순한 H

측정 모델: $\boldsymbol{z}_{\text{gps}} = \boldsymbol{p}_{\text{true}} + \boldsymbol{v}_{\text{gps}}$

명목 예측: $h(\hat{\boldsymbol{x}}) = \hat{\boldsymbol{p}}$

잔차: $\boldsymbol{r} = \boldsymbol{z}_{\text{gps}} - \hat{\boldsymbol{p}} = (\hat{\boldsymbol{p}} + \delta\boldsymbol{p}) - \hat{\boldsymbol{p}} + \boldsymbol{v}_{\text{gps}} = \delta\boldsymbol{p} + \boldsymbol{v}_{\text{gps}}$

H 행렬 (3×15, 15D 상태 순서 $[\delta\boldsymbol{\phi},\ \delta\boldsymbol{p},\ \delta\boldsymbol{v},\ \delta\boldsymbol{b}_g,\ \delta\boldsymbol{b}_a]$):

$$\boldsymbol{H}_{\text{pos}} = \begin{pmatrix} \underbrace{0_{3\times3}}_{\delta\boldsymbol{\phi}} & \underbrace{\boldsymbol{I}_{3\times3}}_{\delta\boldsymbol{p}} & \underbrace{0_{3\times3}}_{\delta\boldsymbol{v}} & \underbrace{0_{3\times3}}_{\delta\boldsymbol{b}_g} & \underbrace{0_{3\times3}}_{\delta\boldsymbol{b}_a} \end{pmatrix}$$

GPS 좌표계

GPS는 보통 ECEF 또는 WGS84 (위도/경도/고도).
로컬 ENU로 변환 후 사용하거나, OpenVINS처럼 처음 위치를 원점으로 삼는다.

GPS 속도 측정

GPS 수신기의 도플러 기반 속도:

$$\boldsymbol{z}_{\text{vel}} = \boldsymbol{v}_{\text{true}} + \boldsymbol{v}_{\text{gps_vel}}$$

명목 예측: $h(\hat{\boldsymbol{x}}) = \hat{\boldsymbol{v}}$

$$\boldsymbol{H}_{\text{vel}} = \begin{pmatrix} 0 & 0 & \boldsymbol{I} & 0 & 0 \end{pmatrix} \quad (3 \times 15)$$

GPS 속도 측정

GPS 수신기의 도플러 기반 속도:

$$\boldsymbol{z}_{\text{vel}} = \boldsymbol{v}_{\text{true}} + \boldsymbol{v}_{\text{gps_vel}}$$

명목 예측: $h(\hat{\boldsymbol{x}}) = \hat{\boldsymbol{v}}$

$$\boldsymbol{H}_{\text{vel}} = \begin{pmatrix} 0 & 0 & \boldsymbol{I} & 0 & 0 \end{pmatrix} \quad (3 \times 15)$$

GPS는 World frame 속도를 준다. 이것이 ESKF의 $\delta \boldsymbol{v}$ 블록에 직접 연결됨.

GPS 노이즈 모델

- 위치 정밀도: RTK $\sim$1cm, 단독 $\sim$3m, 도심 $\sim$10m
- $\boldsymbol{R}_{\text{gps}}$: 수평보다 수직이 크다 (위성 천정 방향만 있어서)

휠 인코더 — Body frame 속도 측정

차동 구동 로봇의 선속도 v_w (Body frame x 방향):

$$\boldsymbol{z}_{\text{wheel}} = \boldsymbol{e}_1^\top \hat{\boldsymbol{R}}^\top \boldsymbol{v}_{\text{true}} + v_w(\text{noise})$$

$\boldsymbol{e}_1 = (1,0,0)^\top$ (body x 방향 단위 벡터)

명목 예측: $h(\hat{\boldsymbol{x}}) = \boldsymbol{e}_1^\top \hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}$

휠 인코더 — Body frame 속도 측정

차동 구동 로봇의 선속도 v_w (Body frame x 방향):

$$\boldsymbol{z}_{\text{wheel}} = \boldsymbol{e}_1^\top \hat{\boldsymbol{R}}^\top \boldsymbol{v}_{\text{true}} + v_w(\text{noise})$$

$\boldsymbol{e}_1 = (1,0,0)^\top$ (body x 방향 단위 벡터)

명목 예측: $h(\hat{\boldsymbol{x}}) = \boldsymbol{e}_1^\top \hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}$ **유도** (right perturbation $\boldsymbol{R} = \hat{\boldsymbol{R}}\,\text{Exp}(\delta\boldsymbol{\phi})$, 1차 근사):

$$\boldsymbol{R}^\top \boldsymbol{v} \approx \hat{\boldsymbol{R}}^\top(\hat{\boldsymbol{v}} + \delta\boldsymbol{v}) + \left[\hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}\right]_\times \delta\boldsymbol{\phi}$$

$$\Rightarrow \quad r = \boldsymbol{e}_1^\top \hat{\boldsymbol{R}}^\top \delta\boldsymbol{v} \;+\; \boldsymbol{e}_1^\top \left[\hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}\right]_\times \delta\boldsymbol{\phi} \;+\; \text{noise}$$

여기서 Adjoint 항등식 $\hat{\boldsymbol{R}}^\top [\hat{\boldsymbol{v}}]_\times \hat{\boldsymbol{R}} = \left[\hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}\right]_\times$ 를 사용.

H 행렬 (1×15):

$$\boldsymbol{H}_{\text{wheel}} = \begin{pmatrix} \boldsymbol{e}_1^\top \left[\hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}\right]_\times & 0 & \boldsymbol{e}_1^\top \hat{\boldsymbol{R}}^\top & 0 & 0 \end{pmatrix}$$

휠 인코더 — Body frame 속도 측정

차동 구동 로봇의 선속도 v_w (Body frame x 방향):

$$\boldsymbol{z}_{\text{wheel}} = \boldsymbol{e}_1^\top \hat{\boldsymbol{R}}^\top \boldsymbol{v}_{\text{true}} + v_w(\text{noise})$$

$\boldsymbol{e}_1 = (1,0,0)^\top$ (body x 방향 단위 벡터)

명목 예측: $h(\hat{\boldsymbol{x}}) = \boldsymbol{e}_1^\top \hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}$ **유도** (right perturbation $\boldsymbol{R} = \hat{\boldsymbol{R}}\,\text{Exp}(\delta\boldsymbol{\phi})$, 1차 근사):

$$\boldsymbol{R}^\top \boldsymbol{v} \approx \hat{\boldsymbol{R}}^\top(\hat{\boldsymbol{v}} + \delta\boldsymbol{v}) + \left[\hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}\right]_\times \delta\boldsymbol{\phi}$$

$$\Rightarrow \quad r = \boldsymbol{e}_1^\top \hat{\boldsymbol{R}}^\top \delta\boldsymbol{v} \;+\; \boldsymbol{e}_1^\top \left[\hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}\right]_\times \delta\boldsymbol{\phi} \;+\; \text{noise}$$

여기서 Adjoint 항등식 $\hat{\boldsymbol{R}}^\top [\hat{\boldsymbol{v}}]_\times \hat{\boldsymbol{R}} = \left[\hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}\right]_\times$ 를 사용.

H 행렬 (1×15):

$$\boldsymbol{H}_{\text{wheel}} = \begin{pmatrix} \boldsymbol{e}_1^\top \left[\hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}\right]_\times & 0 & \boldsymbol{e}_1^\top \hat{\boldsymbol{R}}^\top & 0 & 0 \end{pmatrix}$$

$\delta\boldsymbol{\phi}$ 항이 비자명 — $\left[\hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}\right]$ 때문에 속도 측정만으로도 자세 오차 일부를 관측 가능 (동적 환경에서).

카메라 피처 측정 — MSCKF 원리

MSCKF (Multi-State Constraint KF, OpenVINS의 핵심):

- 3D 피처 위치를 직접 상태로 넣지 않음 (EKF-SLAM 대비 저비용)
- 대신 여러 카메라 pose에서 같은 피처를 보는 기하학적 제약 사용
- 피처를 triangulate → 재투영 잔차 → Update

카메라 피처 측정 — MSCKF 원리

MSCKF (Multi-State Constraint KF, OpenVINS의 핵심):

- 3D 피처 위치를 직접 상태로 넣지 않음 (EKF-SLAM 대비 저비용)
- 대신 여러 카메라 pose에서 같은 피처를 보는 기하학적 제약 사용
- 피처를 triangulate → 재투영 잔차 → Update

재투영 잔차 (pinhole 카메라):

$$\boldsymbol{r}_c = \boldsymbol{z}_c - \pi\Big(\hat{\boldsymbol{R}}_c^\top (\boldsymbol{p}_f - \hat{\boldsymbol{p}}_c)\Big)$$

$\pi(\boldsymbol{p}) = (p_x/p_z,\, p_y/p_z)^\top$: 투영 함수, $\boldsymbol{p}_f$: 피처 3D 위치

카메라 피처 측정 — MSCKF 원리

MSCKF (Multi-State Constraint KF, OpenVINS의 핵심):

- 3D 피처 위치를 직접 상태로 넣지 않음 (EKF-SLAM 대비 저비용)
- 대신 여러 카메라 pose에서 같은 피처를 보는 기하학적 제약 사용
- 피처를 triangulate → 재투영 잔차 → Update

재투영 잔차 (pinhole 카메라):

$$\boldsymbol{r}_c = \boldsymbol{z}_c - \pi\Big(\hat{\boldsymbol{R}}_c^\top (\boldsymbol{p}_f - \hat{\boldsymbol{p}}_c)\Big)$$

$\pi(\boldsymbol{p}) = (p_x/p_z,\, p_y/p_z)^\top$: 투영 함수, $\boldsymbol{p}_f$: 피처 3D 위치

H 행렬
자세 오차 $\delta\boldsymbol{\phi}$와 위치 오차 $\delta\boldsymbol{p}$ 모두에 의존 (비자명):

$$\boldsymbol{H}_c = \boldsymbol{J}_\pi \cdot \hat{\boldsymbol{R}}_c^\top \cdot \Big(-\left[\boldsymbol{p}_f - \hat{\boldsymbol{p}}_c\right]_\times \quad -\boldsymbol{I}\Big)$$

OpenVINS UpdaterMSCKF — H 행렬 구조

ov_msckf/src/update/UpdaterMSCKF.cpp

```
// For each camera pose in the window:
// H_x: Jacobian w.r.t. IMU pose error state
// H_f: Jacobian w.r.t. feature position
Eigen::MatrixXd H_xi = Eigen::MatrixXd::Zero(2, 6);  // 2 pixel residual
H_xi.block(0,0,2,3) = dz_dRci * Rcw * skew_x(pf_w - pi_w); // dz/dphi
H_xi.block(0,3,2,3) = dz_dRci * (-Rcw);                    // dz/dp

// Stack all constraints, project out feature uncertainty (nullspace)
// -> reduced H (jacobian only in pose error space)
```

출처: github.com/rpng/open_vins — GPLv3 라이선스

skew_x = $[\cdot]_\times$, Rcw = $\hat{\boldsymbol{R}}_c^\top$, pf_w = $\boldsymbol{p}_f$

이상값 제거 — Mahalanobis 검정

Update 전 잔차가 통계적으로 말이 되는지 확인:

$$d^2 = \boldsymbol{r}^\top \boldsymbol{S}^{-1} \boldsymbol{r} \overset{H_0}{\sim} \chi^2(\dim \boldsymbol{r})$$

이상값 제거 — Mahalanobis 검정

Update 전 잔차가 통계적으로 말이 되는지 확인:

$$d^2 = \boldsymbol{r}^\top \boldsymbol{S}^{-1} \boldsymbol{r} \overset{H_0}{\sim} \chi^2(\dim \boldsymbol{r})$$

- $d^2 > \chi^2_{0.95}(\dim \boldsymbol{r})$ → 이상값으로 판정, Update 건너뜀
- GPS multipath, 카메라 dynamic object 등 제거에 필수

이상값 제거 — Mahalanobis 검정

Update 전 잔차가 통계적으로 말이 되는지 확인:

$$d^2 = \boldsymbol{r}^\top \boldsymbol{S}^{-1} \boldsymbol{r} \overset{H_0}{\sim} \chi^2(\dim \boldsymbol{r})$$

- $d^2 > \chi^2_{0.95}(\dim \boldsymbol{r})$ → 이상값으로 판정, Update 건너뜀
- GPS multipath, 카메라 dynamic object 등 제거에 필수

실용 임계값

측정 차원	$\chi^2_{0.95}$ 임계값	사용 예
2 (카메라 픽셀)	5.99	OpenVINS 기본값
3 (GPS 위치)	7.81	위성 수신 불량
1 (휠 속도)	3.84	슬립 감지

다중 센서 융합 — 실제 순서

한 타임스텝에서 여러 센서가 동시에 올 때:

1. IMU → Predict (공분산 부풀음)
2. GPS 도착 → Mahalanobis 검정 → Update → Inject & Reset
3. 카메라 피처 → H matrix → Mahalanobis 검정 → Update → Inject & Reset
4. 휠 인코더 → Update (가벼움, 매 step)

다중 센서 융합 — 실제 순서

한 타임스텝에서 여러 센서가 동시에 올 때:

1. IMU → Predict (공분산 부풀음)
2. GPS 도착 → Mahalanobis 검정 → Update → Inject & Reset
3. 카메라 피처 → H matrix → Mahalanobis 검정 → Update → Inject & Reset
4. 휠 인코더 → Update (가벼움, 매 step)

순차 Update vs 일괄 Update
센서들을 독립으로 가정하면 순차(sequential) Update와 일괄 스택(stacked)이 동일.
실제로는 순차가 코드가 간단하고 이상값 제거도 쉽다. OpenVINS도 순차 사용.

레버암(lever arm) — 센서 오프셋 처리

GPS 안테나 위치 $\neq$ IMU 위치:

$$\boldsymbol{p}_{\text{gps}} = \boldsymbol{p}_{\text{IMU}} + \boldsymbol{R}\,\boldsymbol{l}$$

$\boldsymbol{l}$: Body frame 레버암 (calibration)

레버암(lever arm) — 센서 오프셋 처리

GPS 안테나 위치 $\neq$ IMU 위치:

$$\boldsymbol{p}_{\text{gps}} = \boldsymbol{p}_{\text{IMU}} + \boldsymbol{R}\,\boldsymbol{l}$$

$\boldsymbol{l}$: Body frame 레버암 (calibration) $\boldsymbol{l}$를 ESKF 상태에 추가할 수도 있음 (온라인 캘리브레이션). 잔차 선형화:

$$\boldsymbol{r} = \boldsymbol{z}_{\text{gps}} - (\hat{\boldsymbol{p}} + \hat{\boldsymbol{R}}\boldsymbol{l}) = \delta\boldsymbol{p} - \left[\hat{\boldsymbol{R}}\boldsymbol{l}\right]_{\times}\delta\boldsymbol{\phi} + \dots$$

레버암(lever arm) — 센서 오프셋 처리

GPS 안테나 위치 $\neq$ IMU 위치:

$$\boldsymbol{p}_{\text{gps}} = \boldsymbol{p}_{\text{IMU}} + \boldsymbol{R}\,\boldsymbol{l}$$

$\boldsymbol{l}$: Body frame 레버암 (calibration) $\boldsymbol{l}$를 ESKF 상태에 추가할 수도 있음 (온라인 캘리브레이션). 잔차 선형화:

$$\boldsymbol{r} = \boldsymbol{z}_{\text{gps}} - (\hat{\boldsymbol{p}} + \hat{\boldsymbol{R}}\boldsymbol{l}) = \delta\boldsymbol{p} - \left[\hat{\boldsymbol{R}}\boldsymbol{l}\right]_{\times}\,\delta\boldsymbol{\phi} + \dots$$

H 행렬 수정

$$\boldsymbol{H}_{\text{gps+lever}} = \begin{pmatrix} -\left[\hat{\boldsymbol{R}}\boldsymbol{l}\right]_{\times} & \boldsymbol{I} & 0 & 0 & 0 \end{pmatrix}$$

자세 오차도 GPS 잔차에 기여함 → 더 많은 정보 (자세 관측 가능)

카메라-IMU 외적 파라미터 (Extrinsic)

카메라와 IMU는 다른 위치/방향에 있다:

- $\boldsymbol{T}_{BC}$: Body(IMU) frame에서 Camera frame으로의 변환
- $\boldsymbol{p}_{BC}$: 위치 오프셋, $\boldsymbol{R}_{BC}$: 방향 오프셋

카메라-IMU 외적 파라미터 (Extrinsic)

카메라와 IMU는 다른 위치/방향에 있다:

- $\boldsymbol{T}_{BC}$: Body(IMU) frame에서 Camera frame으로의 변환
- $\boldsymbol{p}_{BC}$: 위치 오프셋, $\boldsymbol{R}_{BC}$: 방향 오프셋

캘리브레이션 도구:

- **Kalibr** (`github.com/ethz-asl/kalibr`): IMU + 카메라 외적 캘리브레이션
- ESKF 내부에서 $\boldsymbol{T}_{BC}$를 상태에 추가 → 온라인 추정 가능
- OpenVINS: `state_options.do_calib_camera_pose = true`

카메라-IMU 외적 파라미터 (Extrinsic)

카메라와 IMU는 다른 위치/방향에 있다:

- $\boldsymbol{T}_{BC}$: Body(IMU) frame에서 Camera frame으로의 변환
- $\boldsymbol{p}_{BC}$: 위치 오프셋, $\boldsymbol{R}_{BC}$: 방향 오프셋

캘리브레이션 도구:

- **Kalibr** (`github.com/ethz-asl/kalibr`): IMU + 카메라 외적 캘리브레이션
- ESKF 내부에서 $\boldsymbol{T}_{BC}$를 상태에 추가 → 온라인 추정 가능
- OpenVINS: `state_options.do_calib_camera_pose = true`

실용 팁
외적 파라미터 오차가 크면 ESKF가 발산한다.
초기에 Kalibr로 정확히 캘리브레이션 → 이후 온라인 미세조정.

센서 비교 — 관측 가능 상태

센서	$\delta \boldsymbol{p}$	$\delta \boldsymbol{v}$	$\delta \boldsymbol{\phi}$	$\delta \boldsymbol{b}_g$	$\delta \boldsymbol{b}_a$
GPS 위치	✓				
GPS 속도		✓			
Wheel 속도		✓	†		
카메라 피처	✓		✓		
바로미터 (기압)	†				
자기센서			✓		

† = 부분적/간접적 관측

센서 비교 — 관측 가능 상태

센서	$\delta \boldsymbol{p}$	$\delta \boldsymbol{v}$	$\delta \boldsymbol{\phi}$	$\delta \boldsymbol{b}_g$	$\delta \boldsymbol{b}_a$
GPS 위치	✓				
GPS 속도		✓			
Wheel 속도		✓	†		
카메라 피처	✓		✓		
바로미터 (기압)	†				
자기센서			✓		

† = 부분적/간접적 관측

바이어스 관측 가능성

바이어스 $\delta \boldsymbol{b}_g, \delta \boldsymbol{b}_a$는 센서 하나로는 직접 관측 불가.
다중 센서의 교차 정보 + 시간 + 운동 다양성으로만 추정 가능.

이번 회차 핵심 다섯 개

1. Update 공식은 센서와 무관: $\boldsymbol{r}, \boldsymbol{H}, \boldsymbol{R}, \boldsymbol{K}$, Joseph form
2. 센서마다 $h(\hat{\boldsymbol{x}})$와 $\boldsymbol{H}$만 달라진다
3. GPS: $\boldsymbol{H}_{\text{pos}} = [0, \boldsymbol{I}, 0, 0, 0]$ — 가장 단순
4. 휠/카메라: $\delta\boldsymbol{\phi}$ 블록이 비자명 → 자세 관측에 기여
5. Mahalanobis 검정으로 이상값 제거 — 실용 필수

한 줄 요약
H 행렬 = “센서가 15D 오차 공간의 어느 방향을 보고 있는가”.
방향이 다를수록 바이어스 추정에 도움이 된다.

과제 6 (다음 시간 전)

1. 슬라이드 4의 $\boldsymbol{H}_{\text{pos}}$가 왜 $[\mathbf{0}, \boldsymbol{I}, \mathbf{0}, \mathbf{0}, \mathbf{0}]$인지 잔차 선형화 과정을 직접 유도.
2. 슬라이드 6의 휠 속도 H 행렬에서 $[\hat{\boldsymbol{v}}]_\times$ 항이 등장하는 이유를 수식으로 설명. ($\hat{\boldsymbol{R}}^\top(\hat{\boldsymbol{v}} + \delta\boldsymbol{v}) \approx ?$ 에서 시작.)
3. (선택) OpenVINS `UpdaterHelper.cpp`의 `get_feature_jacobian_representation()` 함수를 읽고, 슬라이드 8의 코드와 대응시켜 설명.
4. (선택) Mahalanobis 임계값을 100으로 두면 어떤 일이 생기는가? χ^2 분포 관점에서 해석.

다음 시간 예고 — 7회차: 관측가능성 + 현장 스택

왜 ESKF가 발산하는가 — 이론과 실전

- 관측가능성 행렬: 상태가 측정값으로 결정될 수 있는가
- 기이한 케이스: 자동차 직선 주행 → $\delta \boldsymbol{b}_g$ 비관측
- NIS (Normalized Innovation Squared) — 필터 건강도 모니터링
- Factor graph로의 이행 (GTSAM, g2o) — smoother와 filter의 차이
- 현장 스택 지도: 어떤 프로젝트가 무엇을 쓰는가

미리 생각해볼 것
로봇이 직선만 달린다면 자이로 z축 바이어스를 추정할 수 있는가?

질문 환영

H 행렬 하나가 15차원 공간의 어느 방향을 가리키는가

7회차: 관측가능성 + Factor Graph

GPS · 휠 · 카메라는 H 행렬 하나로 15D 오차 공간의 서로 다른 방향을 본다.
Mahalanobis 검정으로 이상값 제거.

오늘 (7회차 — ESKF 마무리 + 최적화 세계로의 다리):

- **Observability**: 어떤 상태가 원리적으로 추정 불가인가
- **수치 안정성**: P 양정치성 · Joseph form · Cholesky
- **NIS**: 필터 건강도 진단
- **Factor graph + MAP 최적화**: ESKF 너머의 세계 (8/9회차 사전준비)
- **Marginalization**: 슬라이딩 윈도우 VIO의 핵심
- **현장 스택 지도**: 어떤 프로젝트가 무엇을 쓰는가

Observability — 정의

Observability (관측 가능성)
초기 state x_0를 측정 시퀀스에서 **유일하게** 복원할 수 있는가?

Observability — 정의

Observability (관측 가능성)
초기 state x_0를 측정 시퀀스에서 **유일하게** 복원할 수 있는가?

- Observable: KF가 추정 가능, 분산이 유한하게 수렴
- Unobservable: 분산이 영원히 자란다, 추정 불가능
- 어떤 변수가 unobservable인지 설계 단계에서 알아야 함

선형 시스템의 Observability 행렬

$$\mathcal{O} = \begin{pmatrix} \boldsymbol{H} \\ \boldsymbol{HF} \\ \boldsymbol{HF}^2 \\ \vdots \\ \boldsymbol{HF}^{n-1} \end{pmatrix}$$

- rank$(\mathcal{O}) = n$이면 system은 **completely observable**
- rank $< n$이면 그 차이만큼의 차원이 unobservable
- KF의 가장 이론적 점검 도구

선형 시스템의 Observability 행렬

$$\mathcal{O} = \begin{pmatrix} \boldsymbol{H} \\ \boldsymbol{H}\boldsymbol{F} \\ \boldsymbol{H}\boldsymbol{F}^2 \\ \vdots \\ \boldsymbol{H}\boldsymbol{F}^{n-1} \end{pmatrix}$$

- rank$(\mathcal{O}) = n$이면 system은 **completely observable**
- rank $< n$이면 그 차이만큼의 차원이 unobservable
- KF의 가장 이론적 점검 도구

비선형 시스템 (ESKF): Lie derivative / nonlinear observability 분석.
직관은 동일 — 특정 운동에서 특정 변수가 보이는가.

ESKF에서 unobservable인 것들

- **절대 위치 (GPS 없을 때)**: dead reckoning만 가능, 분산 무한 성장
- **Yaw (자기계/dual-antenna 없고 정지 시)**: gyro 적분 → drift
- **Accel z bias (정지 자세 유지 시)**: 중력과 구분 불가
- **Gyro z bias (yaw 외부 측정 없을 때)**: yaw drift와 1:1 결합

ESKF에서 unobservable인 것들

- **절대 위치 (GPS 없을 때)**: dead reckoning만 가능, 분산 무한 성장
- **Yaw (자기계/dual-antenna 없고 정지 시)**: gyro 적분 → drift
- **Accel z bias (정지 자세 유지 시)**: 중력과 구분 불가
- **Gyro z bias (yaw 외부 측정 없을 때)**: yaw drift와 1:1 결합

교훈
Unobservable 변수는 언젠가는 다른 센서가 잡아줘야 한다.
그 사이는 분산을 허용하고 흘려보내는 것이 정직한 ESKF.

Observability와 Excitation

- “원리적 observability”와 “실제 학습”은 다르다
- 자동차가 항상 직진만 하면: yaw가 사실상 unobservable
- 회전을 한 번이라도 하면 yaw bias의 정보가 들어옴
- 이걸 **excitation** (자극)이라고 부름

Observability와 Excitation

- “원리적 observability”와 “실제 학습”은 다르다
- 자동차가 항상 직진만 하면: yaw가 사실상 unobservable
- 회전을 한 번이라도 하면 yaw bias의 정보가 들어옴
- 이걸 **excitation** (자극)이라고 부름

practical
드론처럼 자세를 자주 바꾸는 시스템은 bias 학습이 빠름.
레일 위 로봇은 영원히 일부 변수를 못 학습.

Mahalanobis distance — 이상치 점검 도구

$$d^2 = \boldsymbol{\nu}^\top \boldsymbol{S}^{-1} \boldsymbol{\nu}$$

- $\boldsymbol{\nu} = \boldsymbol{z} - h(\hat{\boldsymbol{x}})$: residual (innovation)
- $\boldsymbol{S} = \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^\top + \boldsymbol{R}$: 측정 공간 공분산
- d^2: 공분산 가중 거리 — 큰 분산 방향은 “허용”

Mahalanobis distance — 이상치 점검 도구

$$d^2 = \boldsymbol{\nu}^\top \boldsymbol{S}^{-1} \boldsymbol{\nu}$$

- $\boldsymbol{\nu} = \boldsymbol{z} - h(\hat{\boldsymbol{x}})$: residual (innovation)
- $\boldsymbol{S} = \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^\top + \boldsymbol{R}$: 측정 공간 공분산
- d^2: 공분산 가중 거리 — 큰 분산 방향은 “허용”

χ^2 분포
측정이 정상이면 d^2는 자유도 n_z의 χ^2 분포.
임계값 (95% 신뢰): $n_z = 3$일 때 $d^2 \approx 7.81$.

Mahalanobis 기반 거부

매 update step에서:

$$d^2 > d^2_{\text{thresh}} \quad \Rightarrow \quad \text{측정 거부, predict만 진행}$$

- GPS multipath, 위성 dropout
- 카메라 dynamic object (지나가는 차)
- 휠 슬립, 충격

Mahalanobis 기반 거부

매 update step에서:

$$d^2 > d^2_{\text{thresh}} \quad \Rightarrow \quad \text{측정 거부, predict만 진행}$$

- GPS multipath, 위성 dropout
- 카메라 dynamic object (지나가는 차)
- 휠 슬립, 충격

거부하면 그 step은 predict만 진행 → 분산 증가, 그러나 발산 안 함.

OpenVINS 기본값
카메라 피처 $\chi^2_{0.95}(2) = 5.99$ — `chi2_multipler: 1.0`

χ^2 임계값 잡기

- 너무 엄격: 정상 측정도 거부 → 정보 손실 → 발산
- 너무 느슨: 이상치 통과 → 상태 오염 → 발산
- 보통 95% 또는 99% 신뢰도
- $n_z = 3$ at 99%: $d^2 \approx 11.34$

χ^2 임계값 잡기

- 너무 엄격: 정상 측정도 거부 → 정보 손실 → 발산
- 너무 느슨: 이상치 통과 → 상태 오염 → 발산
- 보통 95% 또는 99% 신뢰도
- $n_z = 3$ at 99%: $d^2 \approx 11.34$

practical
실험 데이터로 d^2 분포를 그린 후 결정.
제대로 ESKF가 작동하면 d^2가 정확히 χ^2를 따름 → consistency 점검 도구.

공분산 P의 양정치성

KF 이론: $\boldsymbol{P}$는 항상 **양 준정치** ($\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} \geq 0$).

공분산 P의 양정치성

KF 이론: $\boldsymbol{P}$는 항상 **양 준정치** ($\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} \geq 0$). 실제 코드에서 깨지는 이유:

- Floating point 오차 누적 (특히 $n = 15$ 이상)
- Update의 짧은 form $(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}$는 음의 고유값 만들 수 있음
- $\boldsymbol{R}$이 매우 작을 때 수치 오차 증폭

공분산 P의 양정치성

KF 이론: $\boldsymbol{P}$는 항상 **양 준정치** ($\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} \geq 0$). 실제 코드에서 깨지는 이유:

- Floating point 오차 누적 (특히 $n = 15$ 이상)
- Update의 짧은 form $(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}$는 음의 고유값 만들 수 있음
- $\boldsymbol{R}$이 매우 작을 때 수치 오차 증폭

깨지면 → Cholesky 분해 실패 → ESKF 멈춤

예방책 우선순위
1. **Joseph form** 사용 2. 매 step 대칭화 3. Cholesky 실패 시 shift

P 대칭화 — 가장 싼 보험

매 step 후:

$$\boldsymbol{P} \leftarrow \tfrac{1}{2}(\boldsymbol{P} + \boldsymbol{P}^\top)$$

- Floating point 오차로 생긴 비대칭 제거
- 비용: 거의 없음
- OpenVINS `StateHelper::EKFUpdate()` 내부에서도 사용:

```
P_new = 0.5 * (P_new + P_new.transpose());
```

Joseph form 다시 한 번 — 표준

Update의 두 가지 form:

$$P^{+} = (I - KH)P \quad \text{(짧은 form — 쓰지 말 것)}$$

$$P^{+} = (I - KH)P(I - KH)^{\top} + KRK^{\top} \quad \text{(Joseph form)}$$

Joseph form 다시 한 번 — 표준

Update의 두 가지 form:

$$P^+ = (I - KH)P \quad \text{(짧은 form — 쓰지 말 것)}$$

$$P^+ = (I - KH)P(I - KH)^\top + KRK^\top \quad \text{(Joseph form)}$$

Short form은 **최적 게인일 때만** 정확. 수치 오차가 있으면 음의 고유값 생성. Joseph form은 항상 양정치성을 보장 (대수적으로 P^+가 PSD임을 증명 가능).

규칙: 항상 Joseph form
짧은 form을 쓰면 안 되는 이유가 이것. 3회차부터 반복한 이유.

Cholesky 실패 처리 — identity shift

공분산이 양정치성을 잃으면:

1. 최솟값 고유값 λ_{min} 계산
2. $\boldsymbol{P} \leftarrow \boldsymbol{P} + (-\lambda_{\text{min}} + \epsilon)\boldsymbol{I}$ (identity shift)
3. Cholesky 재시도

Cholesky 실패 처리 — identity shift

공분산이 양정치성을 잃으면:

1. 최솟값 고유값 $\lambda_{\min}$ 계산
2. $\boldsymbol{P} \leftarrow \boldsymbol{P} + (-\lambda_{\min} + \epsilon)\boldsymbol{I}$ (identity shift)
3. Cholesky 재시도

왜 eigenvalue clamp가 아닌가
$\boldsymbol{P}$의 eigenvector는 위치·속도·자세 정보가 혼합되어 있음.
음의 고유값만 clip하면 다른 차원의 분산도 함께 망가짐.
Identity shift는 모든 고유값을 같이 올려 비율 보존.

Time delay와 out-of-order 측정

- GPS는 처리 지연이 100–200ms 일반적
- IMU는 ms 단위, 카메라 exposure + 처리 = 10–50ms
- 측정이 “과거”에 도착하면?

Time delay와 out-of-order 측정

- GPS는 처리 지연이 100–200ms 일반적
- IMU는 ms 단위, 카메라 exposure + 처리 = 10–50ms
- 측정이 “과거”에 도착하면?

대처:

- **Buffered KF**: 과거 state를 저장, 측정 도착 시 재실행
- **Timestamp 무시**: “지금 받았다”고 처리 (단순하지만 오차)
- **Retropropagation**: 과거로 되돌려 update 후 현재로 전파

Time delay와 out-of-order 측정

- GPS는 처리 지연이 100–200ms 일반적
- IMU는 ms 단위, 카메라 exposure + 처리 = 10–50ms
- 측정이 “과거”에 도착하면?

대처:

- **Buffered KF**: 과거 state를 저장, 측정 도착 시 재실행
- **Timestamp 무시**: “지금 받았다”고 처리 (단순하지만 오차)
- **Retropropagation**: 과거로 되돌려 update 후 현재로 전파

OpenVINS: 카메라 이미지 타임스탬프에 맞춰 IMU propagation을 정확히 맞춤.

Multi-rate 센서 처리

IMU 100Hz, 카메라 10Hz, GPS 5Hz — 어떻게?

Multi-rate 센서 처리

IMU 100Hz, 카메라 10Hz, GPS 5Hz — 어떻게? ESKF에서:

1. IMU 도착 시: Predict (명목 + 공분산 전파)
2. 카메라 도착 시: Update (MSCKF residual)
3. GPS 도착 시: Update (위치 residual)

Multi-rate 센서 처리

IMU 100Hz, 카메라 10Hz, GPS 5Hz — 어떻게? ESKF에서:

1. IMU 도착 시: Predict (명목 + 공분산 전파)
2. 카메라 도착 시: Update (MSCKF residual)
3. GPS 도착 시: Update (위치 residual)

중요: **predict의** Δt는 마지막 IMU로부터의 경과 시간.
매번 정확히 계산해야 process noise $\boldsymbol{Q}_d \propto \Delta t$가 맞음.

State 초기화 전략

- Position: GPS 첫 fix를 ENU 원점으로 설정
- Velocity: 0 (정지 가정), 분산 크게 ($\sigma_v^2 \sim 1$ m²/s²)
- 회전 $\hat{\boldsymbol{R}}_0$: 가속도계로 roll/pitch, 자기계/GPS로 yaw
- Bias: 0, 분산 크게 (학습되도록 — 초기 정렬 필요)

State 초기화 전략

- Position: GPS 첫 fix를 ENU 원점으로 설정
- Velocity: 0 (정지 가정), 분산 크게 ($\sigma_v^2 \sim 1$ m²/s²)
- 회전 $\hat{\boldsymbol{R}}_0$: 가속도계로 roll/pitch, 자기계/GPS로 yaw
- Bias: 0, 분산 크게 (학습되도록 — 초기 정렬 필요)

practical
초기 3–5초는 “정렬(alignment)” 구간 — 정지 상태에서 중력 방향 추정.
이 동안 추정값을 써먹지 말 것.
NIS가 $\sim n_z$ 근처에서 안정될 때까지 기다림.

NIS — KF 건강도 진단 도구

Normalized Innovation Squared:

$$\text{NIS}_k = \boldsymbol{r}_k^\top \boldsymbol{S}_k^{-1} \boldsymbol{r}_k \overset{\text{consistent}}{\sim} \chi^2(n_z)$$

- 평균 $= n_z$이면 KF가 **일관성 있다**
- 평균 $\gg n_z$: 모델 부정확, $\boldsymbol{R}$ 또는 $\boldsymbol{Q}$ 너무 작음
- 평균 $\ll n_z$: $\boldsymbol{R}$ 또는 $\boldsymbol{Q}$ 너무 큼 (과도한 노이즈 가정)

NIS — KF 건강도 진단 도구

Normalized Innovation Squared:

$$\text{NIS}_k = \boldsymbol{r}_k^\top \boldsymbol{S}_k^{-1} \boldsymbol{r}_k \overset{\text{consistent}}{\sim} \chi^2(n_z)$$

- 평균 $= n_z$이면 KF가 **일관성 있다**
- 평균 $\gg n_z$: 모델 부정확, $\boldsymbol{R}$ 또는 $\boldsymbol{Q}$ 너무 작음
- 평균 $\ll n_z$: $\boldsymbol{R}$ 또는 $\boldsymbol{Q}$ 너무 큼 (과도한 노이즈 가정)

실험 후 NIS 시계열 plot은 ESKF 튜닝의 **가장 강력한 도구**.
모든 센서에 대해 별도로 그려라.

ESKF 디버깅 — 흔한 함정 7가지

1. **Frame 혼동**: World에 있어야 할 양이 Body, 또는 그 반대
2. **부호 실수**: gravity 방향, 회전 방향 (우/좌 핸드)
3. **시간 동기화**: timestamp 어긋나면 발산
4. **공분산 비대칭**: 매 step 대칭화로 예방
5. **Bias process noise 너무 작음**: bias 학습 중지
6. **측정 분산 R 너무 작음**: 노이즈를 진실로 받아들여 발산
7. **초기 분산 P_0 너무 작음**: 측정이 들어와도 상태 안 움직임

KF 일관성 (Consistency)

KF는 “이 추정의 분산은 P이다”라고 주장한다.
실제 오차 분포가 그 분산과 일치하는가?

KF 일관성 (Consistency)

KF는 “이 추정의 분산은 P이다”라고 주장한다.
실제 오차 분포가 그 분산과 일치하는가?

- **Optimistic** (overconfident): 실제 오차 $>$ 주장 분산 → 위험
- **Pessimistic**: 실제 오차 $<$ 주장 분산 → 안전하지만 비효율

KF 일관성 (Consistency)

KF는 “이 추정의 분산은 $\boldsymbol{P}$이다”라고 주장한다.
실제 오차 분포가 그 분산과 일치하는가?

- **Optimistic** (overconfident): 실제 오차 $>$ 주장 분산 → 위험
- **Pessimistic**: 실제 오차 $<$ 주장 분산 → 안전하지만 비효율

측정법
Truth 있는 시뮬에서: $\mathbb{E}[(\boldsymbol{x}_{\text{true}} - \hat{\boldsymbol{x}})(\cdots)^\top]$와 $\boldsymbol{P}$ 비교.
실제 시스템: NIS로 간접 점검.

ESKF 너머 — Filter vs Smoother

Filter (ESKF)

- 온라인, 실시간
- 과거 측정 재처리 안 함
- 낮은 지연, 낮은 메모리
- 로봇 odometry layer에 적합
- 정확도 한계: 미래 측정 활용 불가

Smoother (Factor Graph)

- 오프라인 또는 fixed-lag online
- 과거 측정도 재처리 (loop closure 등)
- 높은 정확도, 높은 메모리
- 지도 작성, SLAM, 정밀 캘리브레이션
- GTSAM, g2o, Ceres 사용

ESKF 너머 — Filter vs Smoother

Filter (ESKF)

- 온라인, 실시간
- 과거 측정 재처리 안 함
- 낮은 지연, 낮은 메모리
- 로봇 odometry layer에 적합
- 정확도 한계: 미래 측정 활용 불가

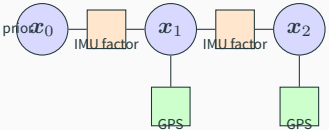
Smoother (Factor Graph)

- 오프라인 또는 fixed-lag online
- 과거 측정도 재처리 (loop closure 등)
- 높은 정확도, 높은 메모리
- 지도 작성, SLAM, 정밀 캘리브레이션
- GTSAM, g2o, Ceres 사용

실제 시스템
Filter (ESKF/iEKF) → 빠른 odometry → Factor Graph (smoother) 입력
예: **FAST-LIO2**(iEKF odometry) + 별도 loop-closure smoother,
LIO-SAM(IMU preintegration + iSAM2 GTSAM smoother)도 동일한 filter+smoother 패턴.

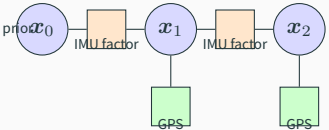
Factor Graph 개념 — 10분 소개

측정값과 상태를 **그래프**로 표현:



Factor Graph 개념 — 10분 소개

측정값과 상태를 **그래프**로 표현:



최적화: 모든 인수의 **잔차 제곱합 최소화** (MAP = maximum a posteriori)
KF의 단일 step 업데이트 $\equiv$ factor graph의 선형 Gaussian 특수 케이스

GTSAM (Georgia Tech Smoothing and Mapping):

- C++ / Python, MIT 라이선스
- IMU, GPS, Camera, LiDAR factor 내장
- Forster 2017 IMU preintegration 내장 (`ImuFactor2`)
- LIO-SAM, VINS-Fusion이 GTSAM 사용

GTSAM과 IMU Preintegration

GTSAM (Georgia Tech Smoothing and Mapping):

- C++ / Python, MIT 라이선스
- IMU, GPS, Camera, LiDAR factor 내장
- Forster 2017 IMU preintegration 내장 (`ImuFactor2`)
- LIO-SAM, VINS-Fusion이 GTSAM 사용

IMU Preintegration Factor
ESKF predict가 매 IMU step 공분산을 전파하듯,
Preintegration factor는 카메라 keyframe 사이 전체 IMU를 하나의 factor로 압축.
바이어스 추정치가 바뀌어도 선형 보정으로 재계산 가능.

현장 스택 지도 — 2025

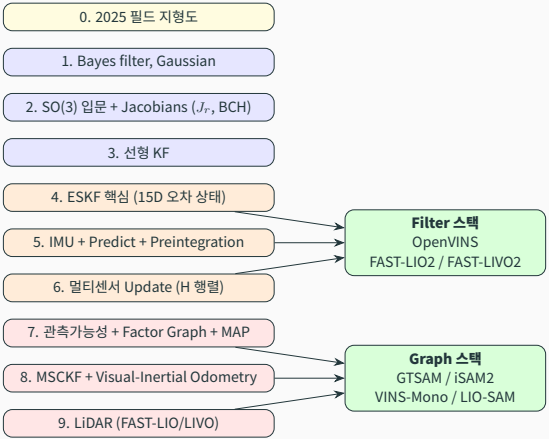
시스템	핵심 기술	레포
OpenVINS	ESKF + MSCKF	`rpng/open_vins`
FAST-LIO2	iEKF on SO(3) + ikd-tree	`hku-mars/FAST_LIO`
FAST-LIVO2	iEKF + LiDAR/Visual	`hku-mars/FAST-LIVO2`
VINS-Mono	Sliding-window factor graph + preint.	`HKUST-Aerial-Robotics/VINS-Mono`
LIO-SAM	IMU preintegration + GTSAM iSAM2	`TixiaoShan/LIO-SAM`
LOAM / LeGO-LOAM	Feature ICP (filter 없음)	`laboshinl/loam_velodyne`
Kalibr	Continuous-time batch (Ceres + B-spline)	`ethz-asl/kalibr`
ROVIO	IEKF on manifold (VIO)	`ethz-asl/rovio`
GTSAM	Factor graph library	`borglab/gtsam`
Basalt	IMU preintegration VIO	`VladyslavUsenko/basalt-headers`

현장 스택 지도 — 2025

시스템	핵심 기술	레포
OpenVINS	ESKF + MSCKF	`rpng/open_vins`
FAST-LIO2	iEKF on SO(3) + ikd-tree	`hku-mars/FAST_LIO`
FAST-LIVO2	iEKF + LiDAR/Visual	`hku-mars/FAST-LIVO2`
VINS-Mono	Sliding-window factor graph + preint.	`HKUST-Aerial-Robotics/VINS-Mono`
LIO-SAM	IMU preintegration + GTSAM iSAM2	`TixiaoShan/LIO-SAM`
LOAM / LeGO-LOAM	Feature ICP (filter 없음)	`laboshinl/loam_velodyne`
Kalibr	Continuous-time batch (Ceres + B-spline)	`ethz-asl/kalibr`
ROVIO	IEKF on manifold (VIO)	`ethz-asl/rovio`
GTSAM	Factor graph library	`borglab/gtsam`
Basalt	IMU preintegration VIO	`VladyslavUsenko/basalt-headers`

공통 패턴: ESKF → 빠른 odometry, Factor graph → 정밀 mapping / loop closure

이 코스 10회를 한 슬라이드로



이 코스가 끝이 아닌 이유

배운 것:

- ESKF — 온라인 filter, IMU + 다중센서, 15D 오차 상태

이 코스가 끝이 아닌 이유

배운 것:

- ESKF — 온라인 filter, IMU + 다중센서, 15D 오차 상태

못 배운 것:

- **Smoother** (RTS, factor graph, GTSAM) — 사후 정제, 정확도 ↑
- **SLAM** (loop closure, mapping) — filter 위의 별개 layer
- **Visual-Inertial Odometry 심화** (MSCKF 유도 전 과정, sliding window)
- **Particle filter** — non-Gaussian, 다중 가설
- **Learning-based** (KalmanNet, deep VIO) — 최근 연구

이 코스가 끝이 아닌 이유

배운 것:

- ESKF — 온라인 filter, IMU + 다중센서, 15D 오차 상태

못 배운 것:

- **Smoother** (RTS, factor graph, GTSAM) — 사후 정제, 정확도 ↑
- **SLAM** (loop closure, mapping) — filter 위의 별개 layer
- **Visual-Inertial Odometry 심화** (MSCKF 유도 전 과정, sliding window)
- **Particle filter** — non-Gaussian, 다중 가설
- **Learning-based** (KalmanNet, deep VIO) — 최근 연구

이 코스는 그 모든 것의 **필수 기초**.

이번 회차 핵심 다섯 개

1. Observability: 어떤 상태는 원리적으로 추정 불가 — 설계 단계에서 확인
2. Joseph form: 항상 사용 — 양정치성 수학적 보장
3. NIS plot: ESKF 튜닝의 가장 강력한 진단 도구
4. Filter vs Smoother: ESKF = odometry layer, factor graph = map layer
5. 실전 스택: OpenVINS / FAST-LIO2 / GTSAM / LIO-SAM 에 이 강의 내용이 모두 들어 있다

학생을 위한 다음 단계 추천

1. **Solà 2017** arXiv:1812.01537 — ESKF 표준 교재, 수식 전부 있음
2. **Barfoot, State Estimation for Robotics** — 이 코스의 정석 교재
3. **Forster 2015/2017** (IMU preintegration) — RSS 2015, TRO 2017
4. OpenVINS 또는 VINS-Mono 코드 전체 읽기 — VIO 입문
5. GTSAM 튜토리얼 + LIO-SAM 코드 — Factor graph 입문
6. EuRoC / KITTI 데이터셋에 OpenVINS 돌려보기 — ATE, RPE 평가

학생을 위한 다음 단계 추천

1. **Solà 2017** arXiv:1812.01537 — ESKF 표준 교재, 수식 전부 있음
2. **Barfoot, State Estimation for Robotics** — 이 코스의 정석 교재
3. **Forster 2015/2017** (IMU preintegration) — RSS 2015, TRO 2017
4. OpenVINS 또는 VINS-Mono 코드 전체 읽기 — VIO 입문
5. GTSAM 튜토리얼 + LIO-SAM 코드 — Factor graph 입문
6. EuRoC / KITTI 데이터셋에 OpenVINS 돌려보기 — ATE, RPE 평가

본인의 첫 state estimation 논문은 이 코스 내용 + 추가 6개월이면 가능.

과제 7 — 최종

1. OpenVINS `Propagator.cpp`를 처음부터 끝까지 읽고, 각 코드 블록이 이 강의 어느 수식에 해당하는지 주석 추가.
2. 1D ESKF를 직접 구현 (명목 상태 + 오차 상태 분리).
   Inject & Reset 후 오차 분산이 실제로 0이 됨을 확인.
3. NIS 시계열을 plot하여 튜닝 전후를 비교.
4. (선택) GTSAM `examples/ImuFactorExample2` 실행 후 ESKF 결과와 비교.
5. (선택) FAST-LIO2의 IEKF 코드와 OpenVINS의 ESKF 코드를 비교 분석.

이 코스에서 기억해야 할 다섯 줄

1. $p(\boldsymbol{x} \mid \boldsymbol{z}) \propto p(\boldsymbol{z} \mid \boldsymbol{x})\, p(\boldsymbol{x})$ — 모든 KF는 이 줄의 가우시안 특수화
2. $\boldsymbol{K} = \boldsymbol{P}\boldsymbol{H}^\top \boldsymbol{S}^{-1}$ — cross-covariance $\times$ 측정 정밀도
3. $\boldsymbol{R} = \hat{\boldsymbol{R}}\,\mathrm{Exp}(\delta\boldsymbol{\phi})$ — ESKF의 핵심: 회전 오차는 $\mathbb{R}^3$에 있다
4. **Joseph form 항상** — $\boldsymbol{P}^+ = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})^\top + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^\top$
5. **NIS로 점검하라** — KF가 자기 분산을 진짜 알고 있는지

여러분이 이제 할 수 있는 것

- OpenVINS `Propagator.cpp` 한 줄씩 “왜”를 답함
- ESKF가 발산하면 어디서 망가졌는지 체크리스트로 진단
- 새 센서가 들어오면 H 행렬을 직접 유도하고 구현
- Observability, bias, outlier를 설계 단계에서 고려
- GTSAM factor graph와 ESKF의 관계를 설명
- 학위논문에서 “KF는 검은 상자”라고 안 쓰게 됨

Part 1을 마치며 — 한 줄 메시지

State estimation은 기초이자 끝이 없는 분야다.

- Kalman 1960. 60년 넘게 살아남은 알고리즘
- 모든 자율주행, 로봇, 항공기, 위성 안에 들어 있음
- 그러나 매 새 센서, 매 새 동역학에서 새로 풀어야
- ESKF는 그 출발점 — Solà 2017을 읽으면 다음이 보인다

Part 1을 마치며 — 한 줄 메시지

State estimation은 기초이자 끝이 없는 분야다.

- Kalman 1960. 60년 넘게 살아남은 알고리즘
- 모든 자율주행, 로봇, 항공기, 위성 안에 들어 있음
- 그러나 매 새 센서, 매 새 동역학에서 새로 풀어야
- ESKF는 그 출발점 — Solà 2017을 읽으면 다음이 보인다

여러분이 이 코스에서 얻은 도구는, 어떤 새 센서, 어떤 새 로봇에서도
“어디서부터 풀까”를 스스로 답하게 해줄 것입니다.

Part 1 마무리 — 그리고 Part 2로

Part 1: 필터 (0–7회차) 끝
여기까지가 **ESKF 중심의 고전 필터 이론** — Solà 2017 / Barfoot 2017의 영역.
OpenVINS·FAST-LIO2 코드를 줄 단위로 읽을 수 있는 토대를 갖췄다.

Part 1 마무리 — 그리고 Part 2로

Part 1: 필터 (0–7회차) 끝
여기까지가 **ESKF 중심의 고전 필터 이론** — Solà 2017 / Barfoot 2017의 영역.
OpenVINS·FAST-LIO2 코드를 줄 단위로 읽을 수 있는 토대를 갖췄다.

Part 2: 최적화 + 인접 영역 (이번 회차 후반 + 8/9회차)

- **MAP + Marginalization** (이번 회차 후반)
- **MSCKF + Visual-Inertial Odometry** (8회차)
- **LiDAR — FAST-LIO/LIVO** (9회차)

공통 키워드: Bell&Cathey 1993의 IEKF≡Gauss-Newton, sparse H, sliding window.

참고문헌(Part 1): Solà 2017 · Barfoot 2017 · Forster 2017 (TRO) · Bell&Cathey 1993.
레포: OpenVINS (`rpng/open_vins`) · FAST-LIO2 (`hku-mars/FAST_LIO`) · GTSAM (`borglab/gtsam`).

7회차 — Part B

필터에서 최적화로

7-A에서 본 것 (방금 마침)
Observability · NIS · Joseph form · Factor graph 입문 · Filter+Smoother 패턴

7-B에서 다룰 것 (이후 40 슬라이드)

- **MAP 추정** = 비선형 최소제곱 (NLLS)
- **Gauss-Newton / LM** — 그리고 IEKF ≡ GN의 등치성
- **Schur complement** — SLAM의 희소 구조 활용
- **Marginalization** — 슬라이딩 윈도우 VIO의 핵심
- **iSAM2** — incremental smoothing, GTSAM의 엔진

이 후반부가 8회차(MSCKF) / 9회차(FAST-LIO IEKF)의 토대.

MAP 추정 — 왜 최적화인가?

Bayes 정리에서 출발

$$p(\boldsymbol{x} \mid \boldsymbol{z}) \;\propto\; p(\boldsymbol{z} \mid \boldsymbol{x})\, p(\boldsymbol{x})$$

MAP 추정 — 왜 최적화인가?

Bayes 정리에서 출발

$$p(\boldsymbol{x} \mid \boldsymbol{z}) \;\propto\; p(\boldsymbol{z} \mid \boldsymbol{x})\, p(\boldsymbol{x})$$

MAP (Maximum A Posteriori) 추정:

$$\boldsymbol{x}^{*} = \arg\max_{\boldsymbol{x}} \; p(\boldsymbol{x} \mid \boldsymbol{z}) = \arg\min_{\boldsymbol{x}} \big[-\log p(\boldsymbol{z} \mid \boldsymbol{x}) - \log p(\boldsymbol{x}) \big]$$

MAP 추정 — 왜 최적화인가?

Bayes 정리에서 출발

$$p(\boldsymbol{x} \mid \boldsymbol{z}) \;\propto\; p(\boldsymbol{z} \mid \boldsymbol{x})\, p(\boldsymbol{x})$$

MAP (Maximum A Posteriori) 추정:

$$\boldsymbol{x}^{*} = \arg\max_{\boldsymbol{x}} \; p(\boldsymbol{x} \mid \boldsymbol{z}) = \arg\min_{\boldsymbol{x}} \big[-\log p(\boldsymbol{z} \mid \boldsymbol{x}) - \log p(\boldsymbol{x}) \big]$$

Gaussian noise 가정 $\Rightarrow -\log p \propto \frac{1}{2}\|r\|^{2}_{\Sigma}$:

$$\boldsymbol{x}^{*} = \arg\min_{\boldsymbol{x}} \sum_{i} \|r_i(\boldsymbol{x})\|^{2}_{\Sigma_i}, \qquad \|r\|^{2}_{\Sigma} \triangleq r^{\top}\Sigma^{-1} r$$

MAP 추정 — 왜 최적화인가?

Bayes 정리에서 출발

$$p(\boldsymbol{x} \mid \boldsymbol{z}) \;\propto\; p(\boldsymbol{z} \mid \boldsymbol{x})\, p(\boldsymbol{x})$$

MAP (Maximum A Posteriori) 추정:

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} \; p(\boldsymbol{x} \mid \boldsymbol{z}) = \arg\min_{\boldsymbol{x}} \big[-\log p(\boldsymbol{z} \mid \boldsymbol{x}) - \log p(\boldsymbol{x}) \big]$$

Gaussian noise 가정 $\Rightarrow -\log p \propto \frac{1}{2}\|r\|^2_{\Sigma}$:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \sum_i \|r_i(\boldsymbol{x})\|^2_{\Sigma_i}, \qquad \|r\|^2_{\Sigma} \triangleq r^\top \Sigma^{-1} r$$

필터 vs. 최적화
ESKF는 이 문제를 순차적으로 한 번씩 풀고 버린다.
MAP 최적화는 전체 윈도우를 한꺼번에 반복 최적화한다.

비선형 최소제곱 문제로의 변환

MAP 목적함수는 **비선형 최소제곱 (Nonlinear Least Squares, NLLS)**:

$$F(\boldsymbol{x}) = \sum_i \|r_i(\boldsymbol{x})\|^2_{\Sigma_i} = \sum_i r_i(\boldsymbol{x})^\top \Sigma_i^{-1} r_i(\boldsymbol{x})$$

비선형 최소제곱 문제로의 변환

MAP 목적함수는 **비선형 최소제곱 (Nonlinear Least Squares, NLLS)**:

$$F(\boldsymbol{x}) = \sum_i \|r_i(\boldsymbol{x})\|^2_{\Sigma_i} = \sum_i r_i(\boldsymbol{x})^\top \Sigma_i^{-1} r_i(\boldsymbol{x})$$

Whitening 변환으로 표준형으로:

$$\tilde{r}_i = \Sigma_i^{-1/2}\, r_i(\boldsymbol{x}) \quad \Rightarrow \quad F(\boldsymbol{x}) = \|\tilde{r}(\boldsymbol{x})\|^2 = \sum_i \|\tilde{r}_i(\boldsymbol{x})\|^2$$

비선형 최소제곱 문제로의 변환

MAP 목적함수는 **비선형 최소제곱 (Nonlinear Least Squares, NLLS)**:

$$F(\boldsymbol{x}) = \sum_i \|r_i(\boldsymbol{x})\|^2_{\Sigma_i} = \sum_i r_i(\boldsymbol{x})^\top \Sigma_i^{-1} r_i(\boldsymbol{x})$$

Whitening 변환으로 표준형으로:

$$\tilde{r}_i = \Sigma_i^{-1/2}\, r_i(\boldsymbol{x}) \quad \Rightarrow \quad F(\boldsymbol{x}) = \|\tilde{r}(\boldsymbol{x})\|^2 = \sum_i \|\tilde{r}_i(\boldsymbol{x})\|^2$$

SLAM/VIO에서의 residuals r_i

- IMU preintegration residual
- 카메라 reprojection residual
- Wheel odometry residual
- Prior (marginalization으로 생성된 prior factor)

비선형 최소제곱 문제로의 변환

MAP 목적함수는 **비선형 최소제곱 (Nonlinear Least Squares, NLLS)**:

$$F(\boldsymbol{x}) = \sum_i \|r_i(\boldsymbol{x})\|^2_{\Sigma_i} = \sum_i r_i(\boldsymbol{x})^\top \Sigma_i^{-1} r_i(\boldsymbol{x})$$

Whitening 변환으로 표준형으로:

$$\tilde{r}_i = \Sigma_i^{-1/2}\, r_i(\boldsymbol{x}) \quad \Rightarrow \quad F(\boldsymbol{x}) = \|\tilde{r}(\boldsymbol{x})\|^2 = \sum_i \|\tilde{r}_i(\boldsymbol{x})\|^2$$

SLAM/VIO에서의 residuals r_i

- IMU preintegration residual
- 카메라 reprojection residual
- Wheel odometry residual
- Prior (marginalization으로 생성된 prior factor)

$\boldsymbol{x}$ = 모든 keyframe poses + 속도 + IMU bias + landmarks (특징점).

Gauss-Newton 알고리즘

핵심 아이디어: $r_i(\boldsymbol{x} + \delta\boldsymbol{x}) \approx r_i(\boldsymbol{x}) + J_i\,\delta\boldsymbol{x}$ (1차 Taylor)

Gauss-Newton 알고리즘

핵심 아이디어: $r_i(\boldsymbol{x}+\delta\boldsymbol{x}) \approx r_i(\boldsymbol{x}) + J_i\,\delta\boldsymbol{x}$ (1차 Taylor)

$$\begin{aligned} F(\boldsymbol{x}+\delta\boldsymbol{x}) &\approx \|\tilde{r} + \tilde{J}\,\delta\boldsymbol{x}\|^2 \\ &= \delta\boldsymbol{x}^\top \underbrace{(\tilde{J}^\top \tilde{J})}_{H}\,\delta\boldsymbol{x} + 2\,\underbrace{\tilde{r}^\top \tilde{J}}_{b^\top}\,\delta\boldsymbol{x} + \text{const} \end{aligned}$$

Gauss-Newton 알고리즘

핵심 아이디어: $r_i(\boldsymbol{x} + \delta\boldsymbol{x}) \approx r_i(\boldsymbol{x}) + J_i\,\delta\boldsymbol{x}$ (1차 Taylor)

$$\begin{aligned} F(\boldsymbol{x} + \delta\boldsymbol{x}) &\approx \|\tilde{r} + \tilde{J}\,\delta\boldsymbol{x}\|^2 \\ &= \delta\boldsymbol{x}^\top \underbrace{(\tilde{J}^\top \tilde{J})}_{H}\,\delta\boldsymbol{x} + 2\,\underbrace{\tilde{r}^\top \tilde{J}}_{b^\top}\,\delta\boldsymbol{x} + \text{const} \end{aligned}$$

Normal Equation

$$H\,\delta\boldsymbol{x} = -b, \qquad H = \sum_i J_i^\top \Sigma_i^{-1} J_i, \quad b = \sum_i J_i^\top \Sigma_i^{-1} r_i(\boldsymbol{x})$$

H: **Information matrix** (positive semi-definite)

Gauss-Newton 알고리즘

핵심 아이디어: $r_i(\boldsymbol{x}+\delta\boldsymbol{x}) \approx r_i(\boldsymbol{x}) + J_i\,\delta\boldsymbol{x}$ (1차 Taylor)

$$\begin{aligned} F(\boldsymbol{x}+\delta\boldsymbol{x}) &\approx \|\tilde{r} + \tilde{J}\,\delta\boldsymbol{x}\|^2 \\ &= \delta\boldsymbol{x}^\top \underbrace{(\tilde{J}^\top \tilde{J})}_{H}\,\delta\boldsymbol{x} + 2\,\underbrace{\tilde{r}^\top \tilde{J}}_{b^\top}\,\delta\boldsymbol{x} + \text{const} \end{aligned}$$

Normal Equation

$$H\,\delta\boldsymbol{x} = -b, \qquad H = \sum_i J_i^\top \Sigma_i^{-1} J_i, \quad b = \sum_i J_i^\top \Sigma_i^{-1} r_i(\boldsymbol{x})$$

H: **Information matrix** (positive semi-definite)

Update (manifold 상의 retraction):

$$\boldsymbol{x} \;\leftarrow\; \boldsymbol{x} \boxplus (-H^{-1}b)$$

$\boxplus$: SO(3)/SE(3) 위에서는 Exp map을 통해 적용.

수렴

(E)KF Update ≡ Gauss-Newton 한 스텝

핵심 결과 — Bell & Cathey (IEEE TAC, 1993)
“The iterated Kalman filter update as a Gauss-Newton method”
IEKF의 한 iteration은 MAP cost에 대한 정확히 한 번의 Gauss-Newton step과 동치이다.
따라서 **표준 EKF/ESKF update = Gauss-Newton 1회 iteration**, $\hat{x}^-$에서 선형화한 결과.

(E)KF Update ≡ Gauss-Newton 한 스텝

핵심 결과 — Bell & Cathey (IEEE TAC, 1993)
"The iterated Kalman filter update as a Gauss-Newton method"
IEKF의 한 iteration은 MAP cost에 대한 정확히 한 번의 Gauss-Newton step과 동치이다.
따라서 **표준 EKF/ESKF update = Gauss-Newton 1회 iteration**, $\hat{x}^-$에서 선형화한 결과.

한 스텝 MAP의 cost (prior $\hat{\boldsymbol{x}}^-$, $\boldsymbol{P}^-$, 측정 $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{H}\boldsymbol{x},\ \boldsymbol{R})$):

$$F(\boldsymbol{x}) = \tfrac{1}{2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}^-\|^2_{(\boldsymbol{P}^-)^{-1}} + \tfrac{1}{2}\|\boldsymbol{z} - h(\boldsymbol{x})\|^2_{\boldsymbol{R}^{-1}}$$

(E)KF Update ≡ Gauss-Newton 한 스텝

핵심 결과 — Bell & Cathey (IEEE TAC, 1993)
"The iterated Kalman filter update as a Gauss-Newton method"
IEKF의 한 iteration은 MAP cost에 대한 정확히 한 번의 Gauss-Newton step과 동치이다.
따라서 **표준 EKF/ESKF update = Gauss-Newton 1회 iteration**, $\hat{x}^-$에서 선형화한 결과.

한 스텝 MAP의 cost (prior $\hat{\boldsymbol{x}}^-$, $\boldsymbol{P}^-$, 측정 $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{H}\boldsymbol{x},\ \boldsymbol{R})$):

$$F(\boldsymbol{x}) = \tfrac{1}{2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}^-\|^2_{(\boldsymbol{P}^-)^{-1}} + \tfrac{1}{2}\|\boldsymbol{z} - h(\boldsymbol{x})\|^2_{\boldsymbol{R}^{-1}}$$

$\hat{\boldsymbol{x}}^-$ 주변 1차 선형화 후 normal equation을 풀면:

$$\delta\boldsymbol{x}^* = (\boldsymbol{P}^{-1} + \boldsymbol{H}^\top\boldsymbol{R}^{-1}\boldsymbol{H})^{-1}\boldsymbol{H}^\top\boldsymbol{R}^{-1}(\boldsymbol{z} - h(\hat{\boldsymbol{x}}^-)) = \boldsymbol{K}\,(\boldsymbol{z} - h(\hat{\boldsymbol{x}}^-))$$

(E)KF Update ≡ Gauss-Newton 한 스텝

핵심 결과 — Bell & Cathey (IEEE TAC, 1993)
"The iterated Kalman filter update as a Gauss-Newton method"
IEKF의 한 iteration은 MAP cost에 대한 정확히 한 번의 Gauss-Newton step과 동치이다.
따라서 **표준 EKF/ESKF update = Gauss-Newton 1회 iteration**, $\hat{x}^-$에서 선형화한 결과.

한 스텝 MAP의 cost (prior $\hat{\boldsymbol{x}}^-$, $\boldsymbol{P}^-$, 측정 $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{H}\boldsymbol{x},\ \boldsymbol{R})$):

$$F(\boldsymbol{x}) = \tfrac{1}{2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}^-\|^2_{(\boldsymbol{P}^-)^{-1}} + \tfrac{1}{2}\|\boldsymbol{z} - h(\boldsymbol{x})\|^2_{\boldsymbol{R}^{-1}}$$

$\hat{\boldsymbol{x}}^-$ 주변 1차 선형화 후 normal equation을 풀면:

$$\delta\boldsymbol{x}^* = (\boldsymbol{P}^{-1} + \boldsymbol{H}^\top\boldsymbol{R}^{-1}\boldsymbol{H})^{-1}\boldsymbol{H}^\top\boldsymbol{R}^{-1}(\boldsymbol{z} - h(\hat{\boldsymbol{x}}^-)) = \boldsymbol{K}\,(\boldsymbol{z} - h(\hat{\boldsymbol{x}}^-))$$

함의 — 이 코스 전체를 관통하는 한 줄
ESKF ⊂ Iterated EKF ⊂ Gauss-Newton on MAP ⊂ Factor Graph 최적화.
필터와 스무더는 다른 알고리즘이 아니라 같은 cost를 푸는 다른 스케줄이다.

참고: Bell & Cathey 1993 · Skoglund, Hendeby, Axehill 2015 (FUSION) · Sibley, Sukhatme, Matthies 2006 (RSS) · Barfoot 2017 §4.2.

ESKF vs. Iterated ESKF — 왜 FAST-LIO가 IEKF를 쓰는가

표준 ESKF (1-step)

- Linearize 한 번 at $\hat{\boldsymbol{x}}^-$
- Update 후 종료
- 측정 비선형성이 심하면 편향
- $h(\hat{\boldsymbol{x}}^- + \boldsymbol{K}\nu) \neq \boldsymbol{z}$ 가능

⇒ Gauss-Newton의 **1차 iteration**

Iterated ESKF (IEKF)

- Re-linearize at 갱신된 $\hat{\boldsymbol{x}}^{(l)}$
- 수렴할 때까지 반복
- 비선형 측정에 더 정확
- LiDAR point-to-plane에 필수

⇒ **전체 Gauss-Newton iteration**

ESKF vs. Iterated ESKF — 왜 FAST-LIO가 IEKF를 쓰는가

표준 ESKF (1-step)

- Linearize 한 번 at $\hat{\boldsymbol{x}}^-$
- Update 후 종료
- 측정 비선형성이 심하면 편향
- $h(\hat{\boldsymbol{x}}^- + \boldsymbol{K}\nu) \neq \boldsymbol{z}$ 가능

⇒ Gauss-Newton의 **1차 iteration**

Iterated ESKF (IEKF)

- Re-linearize at 갱신된 $\hat{\boldsymbol{x}}^{(l)}$
- 수렴할 때까지 반복
- 비선형 측정에 더 정확
- LiDAR point-to-plane에 필수

⇒ **전체 Gauss-Newton iteration**

IEKF iteration l — manifold 형태 (FAST-LIO2 Eq. 18)
표기: $\tilde{\boldsymbol{x}}^{(l)} \triangleq \hat{\boldsymbol{x}}^{(l)} \boxminus \hat{\boldsymbol{x}}^-$, $r^{(l)} \triangleq \boldsymbol{z} - h(\hat{\boldsymbol{x}}^{(l)})$, $\boldsymbol{A}^{(l)} \triangleq (\boldsymbol{P}^-)^{-1} + \boldsymbol{H}^{(l)\top}\boldsymbol{R}^{-1}\boldsymbol{H}^{(l)}$.

$$\delta\boldsymbol{x}^{(l)} \;=\; \big(\boldsymbol{A}^{(l)}\big)^{-1}\Big(\, \boldsymbol{H}^{(l)\top}\boldsymbol{R}^{-1}\, r^{(l)} \;-\; (\boldsymbol{P}^-)^{-1}\,\tilde{\boldsymbol{x}}^{(l)} \,\Big)$$

$$\hat{\boldsymbol{x}}^{(l+1)} = \hat{\boldsymbol{x}}^{(l)} \;\boxplus\; \delta\boldsymbol{x}^{(l)}$$

수렴 조건: $\|\delta\boldsymbol{x}^{(l)}\|_\infty < \epsilon$. $l = 0$에서 $\tilde{\boldsymbol{x}}^{(0)} = \mathbf{0}$이므로 표준 EKF update와 정확히 일치.

ESKF vs. Iterated ESKF — 왜 FAST-LIO가 IEKF를 쓰는가

표준 ESKF (1-step)

- Linearize 한 번 at $\hat{\boldsymbol{x}}^{-}$
- Update 후 종료
- 측정 비선형성이 심하면 편향
- $h(\hat{\boldsymbol{x}}^{-} + \boldsymbol{K}\nu) \neq \boldsymbol{z}$ 가능

⇒ Gauss-Newton의 **1차 iteration**

Iterated ESKF (IEKF)

- Re-linearize at 갱신된 $\hat{\boldsymbol{x}}^{(l)}$
- 수렴할 때까지 반복
- 비선형 측정에 더 정확
- LiDAR point-to-plane에 필수

⇒ **전체 Gauss-Newton iteration**

IEKF iteration l — manifold 형태 (FAST-LIO2 Eq. 18)
표기: $\tilde{\boldsymbol{x}}^{(l)} \triangleq \hat{\boldsymbol{x}}^{(l)} \boxminus \hat{\boldsymbol{x}}^{-}$, $r^{(l)} \triangleq \boldsymbol{z} - h(\hat{\boldsymbol{x}}^{(l)})$, $\boldsymbol{A}^{(l)} \triangleq (\boldsymbol{P}^{-})^{-1} + \boldsymbol{H}^{(l)\top}\boldsymbol{R}^{-1}\boldsymbol{H}^{(l)}$.

$$\delta\boldsymbol{x}^{(l)} \;=\; \big(\boldsymbol{A}^{(l)}\big)^{-1}\Big(\, \boldsymbol{H}^{(l)\top}\boldsymbol{R}^{-1}\, r^{(l)} \;-\; (\boldsymbol{P}^{-})^{-1}\,\tilde{\boldsymbol{x}}^{(l)} \,\Big)$$

$$\hat{\boldsymbol{x}}^{(l+1)} = \hat{\boldsymbol{x}}^{(l)} \;\boxplus\; \delta\boldsymbol{x}^{(l)}$$

수렴 조건: $\|\delta\boldsymbol{x}^{(l)}\|_{\infty} < \epsilon$. $l = 0$에서 $\tilde{\boldsymbol{x}}^{(0)} = \mathbf{0}$이므로 표준 EKF update와 정확히 일치.

현장 의미
FAST-LIO/2/LIVO/LIVO2가 EKF 대신 IEKF를 쓰는 이유 = 측정 nonlinearity가 큰 LiDAR에서 Gauss-Newton 수렴 정확도가 필요하기 때문.

Levenberg-Marquardt (LM)

Gauss-Newton의 문제
H가 singular에 가깝거나 초기값이 나쁠 때 δx가 발산

Levenberg-Marquardt (LM)

Gauss-Newton의 문제
H가 singular에 가깝거나 초기값이 나쁠 때 $\delta \boldsymbol{x}$가 발산

LM 수정: 대각 damping 추가

$$H_\lambda \, \delta \boldsymbol{x} = -b, \qquad H_\lambda = H + \lambda \, \mathrm{diag}(H)$$

Levenberg-Marquardt (LM)

Gauss-Newton의 문제
H가 singular에 가깝거나 초기값이 나쁠 때 $\delta \boldsymbol{x}$가 발산

LM 수정: 대각 damping 추가

$$H_\lambda \, \delta \boldsymbol{x} = -b, \qquad H_\lambda = H + \lambda \, \mathrm{diag}(H)$$

λ 크면

- $H_\lambda \approx \lambda \, \mathrm{diag}(H)$
- Gradient descent 방향
- 느리지만 안정적
- 수렴 보장 강함

λ 작으면

- $H_\lambda \approx H$ (Gauss-Newton)
- 빠른 수렴
- 초기값 민감
- quadratic convergence

Levenberg-Marquardt (LM)

Gauss-Newton의 문제
H가 singular에 가깝거나 초기값이 나쁠 때 $\delta \boldsymbol{x}$가 발산

LM 수정: 대각 damping 추가

$$H_\lambda \, \delta \boldsymbol{x} = -b, \qquad H_\lambda = H + \lambda \, \mathrm{diag}(H)$$

λ 크면

- $H_\lambda \approx \lambda \, \mathrm{diag}(H)$
- Gradient descent 방향
- 느리지만 안정적
- 수렴 보장 강함

λ 작으면

- $H_\lambda \approx H$ (Gauss-Newton)
- 빠른 수렴
- 초기값 민감
- quadratic convergence

λ 업데이트: F 감소 $\rightarrow$ λ 줄임 (GN으로); F 증가 $\rightarrow$ λ 늘림 (GD로)

수렴 기준: $\|r\|$ 변화 $< \epsilon_1$, $\|\delta \boldsymbol{x}\| < \epsilon_2$, $\|b\|_\infty < \epsilon_3$

실용

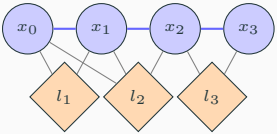
SLAM의 희소 구조

Factor graph 구조:

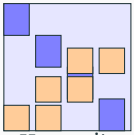
poses $x_0, x_1, \ldots, x_n$ + landmarks $l_1, \ldots, l_m$

각 factor는 2∼3개 변수에만 연결

$\Rightarrow$ Jacobian J: **매우 희소 (sparse)**



$H = J^\top J$: **sparse symmetric**



H sparsity

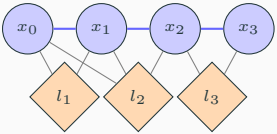
SLAM의 희소 구조

Factor graph 구조:

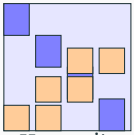
poses $x_0, x_1, \ldots, x_n$ + landmarks $l_1, \ldots, l_m$

각 factor는 2~3개 변수에만 연결

$\Rightarrow$ Jacobian J: **매우 희소 (sparse)**



$H = J^\top J$: **sparse symmetric**



H sparsity

희소성의 중요성

Full matrix inverse $O(n^3)$에서 chain topology에서는 sparse solver $O(n)$으로 감소.

100개 keyframe: dense $\rightarrow$ ~10^6 ops, sparse $\rightarrow$ ~10^3 ops (1000배 차이!)

Sparse Cholesky Factorization

Symmetric positive definite H에 대해:

$$H = L\,L^{\top} \qquad (L : \text{lower triangular})$$

Sparse Cholesky Factorization

Symmetric positive definite H에 대해:

$$H = L\,L^{\top} \qquad (L : \text{lower triangular})$$

$H\,\delta\boldsymbol{x} = -b$ 풀기:

1. $L\,y = -b$ (forward substitution)
2. $L^{\top}\,\delta\boldsymbol{x} = y$ (backward substitution)

Sparse Cholesky Factorization

Symmetric positive definite H에 대해:

$$H = L\,L^{\top} \qquad (L : \text{lower triangular})$$

$H\,\delta\boldsymbol{x} = -b$ 풀기:

1. $L\,y = -b$ (forward substitution)
2. $L^{\top}\,\delta\boldsymbol{x} = y$ (backward substitution)

Fill-in 문제
Cholesky 인수분해 과정에서 0이었던 위치가 nonzero가 되는 현상
$\rightarrow$ variable ordering이 fill-in 양에 결정적으로 영향

Sparse Cholesky Factorization

Symmetric positive definite H에 대해:

$$H = L\,L^{\top} \qquad (L : \text{lower triangular})$$

$H\,\delta\boldsymbol{x} = -b$ 풀기:

1. $L\,y = -b$ (forward substitution)
2. $L^{\top}\,\delta\boldsymbol{x} = y$ (backward substitution)

Fill-in 문제
Cholesky 인수분해 과정에서 0이었던 위치가 nonzero가 되는 현상
→ variable ordering이 fill-in 양에 결정적으로 영향

Ordering 알고리즘

- **COLAMD**: Column Approximate Minimum Degree
- **AMD**: Approximate Minimum Degree
- GTSAM: COLAMD 기본 사용

구현 라이브러리

- `Eigen::SimplicialLLT<>`
- `CHOLMOD` (SuiteSparse)
- GTSAM 내장 linear solver

Schur Complement — 블록 구조 활용

SLAM의 전형적 H 구조 (poses p + landmarks l):

$$H = \begin{pmatrix} H_{pp} & H_{pl} \\ H_{lp} & H_{ll} \end{pmatrix}, \qquad b = \begin{pmatrix} b_p \\ b_l \end{pmatrix}$$

Schur Complement — 블록 구조 활용

SLAM의 전형적 H 구조 (poses p + landmarks l):

$$H = \begin{pmatrix} H_{pp} & H_{pl} \\ H_{lp} & H_{ll} \end{pmatrix}, \qquad b = \begin{pmatrix} b_p \\ b_l \end{pmatrix}$$

핵심 관찰: H_{ll}은 block-diagonal (각 landmark는 독립)
$\Rightarrow H_{ll}^{-1}$은 각 블록 역행렬의 집합 = 매우 저렴!

Schur Complement — 블록 구조 활용

SLAM의 전형적 H 구조 (poses p + landmarks l):

$$H = \begin{pmatrix} H_{pp} & H_{pl} \\ H_{lp} & H_{ll} \end{pmatrix}, \qquad b = \begin{pmatrix} b_p \\ b_l \end{pmatrix}$$

핵심 관찰: H_{ll}은 block-diagonal (각 landmark는 독립)
$\Rightarrow H_{ll}^{-1}$은 각 블록 역행렬의 집합 = 매우 저렴!

Schur Complement 기법

$$S \;=\; H_{pp} - H_{pl}\, H_{ll}^{-1}\, H_{lp} \quad \text{(reduced system, pose 차원만)}$$

1. Reduced system 풀기: $S\,\delta p = -b_p + H_{pl}\, H_{ll}^{-1}\, b_l$
2. Landmark back-substitution: $\delta l = H_{ll}^{-1}(b_l - H_{lp}\,\delta p)$

Schur Complement — 블록 구조 활용

SLAM의 전형적 H 구조 (poses p + landmarks l):

$$H = \begin{pmatrix} H_{pp} & H_{pl} \\ H_{lp} & H_{ll} \end{pmatrix}, \qquad b = \begin{pmatrix} b_p \\ b_l \end{pmatrix}$$

핵심 관찰: H_{ll}은 block-diagonal (각 landmark는 독립)
$\Rightarrow H_{ll}^{-1}$은 각 블록 역행렬의 집합 = 매우 저렴!

Schur Complement 기법

$$S \;=\; H_{pp} - H_{pl}\, H_{ll}^{-1}\, H_{lp} \quad \text{(reduced system, pose 차원만)}$$

1. Reduced system 풀기: $S\,\delta p = -b_p + H_{pl}\, H_{ll}^{-1}\, b_l$
2. Landmark back-substitution: $\delta l = H_{ll}^{-1}(b_l - H_{lp}\,\delta p)$

효율
1000개 landmarks, 100개 poses: landmark eliminate 후 100×100 크기 선형계만 풀면 됨.
VSLAM/SfM (Bundle Adjustment)의 표준 기법.

SAM의 문제 (Batch)
새 factor 추가마다 full re-factorization $\rightarrow O(n^3)$:
100 keyframes마다 처음부터 다시 Cholesky 분해 = 실시간 불가

SAM의 문제 (Batch)
새 factor 추가마다 full re-factorization $\rightarrow O(n^3)$:
100 keyframes마다 처음부터 다시 Cholesky 분해 = 실시간 불가

iSAM (Kaess et al., 2008):

- Cholesky의 rank-1 update 트릭 활용
- 주기적 reordering으로 fill-in 관리
- $O(n)$ amortized cost

iSAM / iSAM2 — Incremental Smoothing and Mapping

SAM의 문제 (Batch)
새 factor 추가마다 full re-factorization $\rightarrow O(n^3)$:
100 keyframes마다 처음부터 다시 Cholesky 분해 = 실시간 불가

iSAM (Kaess et al., 2008):

- Cholesky의 rank-1 update 트릭 활용
- 주기적 reordering으로 fill-in 관리
- $O(n)$ amortized cost

iSAM2 (Kaess et al., 2012):

- **Bayes tree** 자료구조: Cholesky factor를 tree로 인코딩
- 새 factor가 추가되면 영향받는 clique만 재계산
- 실용적으로 단일 factor 추가 = $O(1)$ (sparse graph)
- Wildfire 알고리즘: 변경 전파 범위를 tree로 추적

iSAM / iSAM2 — Incremental Smoothing and Mapping

SAM의 문제 (Batch)
새 factor 추가마다 full re-factorization $\rightarrow O(n^3)$:
100 keyframes마다 처음부터 다시 Cholesky 분해 = 실시간 불가

iSAM (Kaess et al., 2008):

- Cholesky의 rank-1 update 트릭 활용
- 주기적 reordering으로 fill-in 관리
- $O(n)$ amortized cost

iSAM2 (Kaess et al., 2012):

- **Bayes tree** 자료구조: Cholesky factor를 tree로 인코딩
- 새 factor가 추가되면 영향받는 clique만 재계산
- 실용적으로 단일 factor 추가 = $O(1)$ (sparse graph)
- Wildfire 알고리즘: 변경 전파 범위를 tree로 추적

실제 시스템
OpenVINS: EKF 기반, iSAM 미사용
GTSAM 기반 VIO (e.g., Kimera-VIO): iSAM2 핵심 사용

GTSAM 기본 구조


```cpp
#include <gtsam/nonlinear/NonlinearFactorGraph.h>
#include <gtsam/nonlinear/Values.h>
#include <gtsam/nonlinear/LevenbergMarquardtOptimizer.h>
#include <gtsam/inference/Symbol.h>
#include <gtsam/geometry/Pose3.h>

NonlinearFactorGraph graph;
Values initialEstimate;

// 첫번째에  pose prior 추가 (6-DoF: roll,pitch,yaw,x,y,z)
auto noise = noiseModel::Diagonal::Sigmas(
    (Vector(6) << 0.1, 0.1, 0.1, 0.3, 0.3, 0.3).finished());
graph.addPrior(Symbol('x', 0), Pose3(), noise);
initialEstimate.insert(Symbol('x', 0), Pose3());

// 최적화
LevenbergMarquardtParams params;
params.verbosity = LevenbergMarquardtParams::VALUES;
LevenbergMarquardtOptimizer opt(graph, initialEstimate, params);
Values result = opt.optimize();
```

GTSAM IMU Factor 예시


```cpp
#include <gtsam/navigation/CombinedImuFactor.h>

// 파라미터설정  (gravity = 9.81, NED or ENU)
auto params = PreintegrationCombinedParams::MakeSharedU(9.81);
params->gyroscopeCovariance     = sigma_gyr * I_3x3;
params->accelerometerCovariance = sigma_acc * I_3x3;
params->biasAccCovariance       = sigma_ba  * I_3x3;

// IMU preintegration (keyframe i -> j 사이)
PreintegratedCombinedMeasurements pim(params, imuBias::ConstantBias());
while (have_imu_data) {
    pim.integrateMeasurement(accel, gyro, dt);
}

// CombinedImuFactor: pose + vel + bias 개6 변수를동시에연결
graph.add(CombinedImuFactor(
    X(i), V(i), X(j), V(j), B(i), B(j), pim));
```

Combined IMU Factor의 특징

- IMU preintegration residual (9-dim: position, velocity, rotation)
- Bias random walk constraint를 하나의 factor로 통합
- Bias 공분산 propagation 포함

GTSAM — Between Factor & Pose Graph


```cpp
#include <gtsam/slam/BetweenFactor.h>

// 두 pose 사이의 relative transform (odometry, loop closure)
auto odomNoise = noiseModel::Diagonal::Sigmas(
    (Vector(6) << 0.05, 0.05, 0.05, 0.1, 0.1, 0.1).finished());

Pose3 relativePose = pose_i.between(pose_j);  // T_i^{-1} * T_j
graph.add(BetweenFactor<Pose3>(
    Symbol('x', i), Symbol('x', j), relativePose, odomNoise));

// 초기값설정및최적화
for (int k = 0; k <= N; k++)
    initialEstimate.insert(Symbol('x', k), initial_poses[k]);

ISAM2 isam;                    // iSAM2 객체
isam.update(graph, initialEstimate);
Values current = isam.calculateEstimate();
```

iSAM2 incremental update

`isam.update()` 호출마다 Bayes tree를 증분 업데이트.

MAP 최적화 — 수렴 분석

수렴 보장 조건

- H가 positive definite (관측 가능)
- 초기값이 수렴 반경 내
- Residual이 smooth (미분 가능)
- 충분한 iteration

실패 상황

- 잘못된 루프 클로저
- 초기값 불량 (global minimum 못 찾음)
- Degenerate motion (관측 불가 방향)
- 수치 오류 (scale 차이)

MAP 최적화 — 수렴 분석

수렴 보장 조건

- H가 positive definite (관측 가능)
- 초기값이 수렴 반경 내
- Residual이 smooth (미분 가능)
- 충분한 iteration

실패 상황

- 잘못된 루프 클로저
- 초기값 불량 (global minimum 못 찾음)
- Degenerate motion (관측 불가 방향)
- 수치 오류 (scale 차이)

Robustness: Robust Cost Functions

Gaussian 대신 **Huber**, **Cauchy**, **GM** 등 robust loss:

$$\rho(r) = \begin{cases} \frac{1}{2}r^2 & |r| \le k \quad \text{(quadratic)} \\ k|r| - \frac{k^2}{2} & |r| > k \quad \text{(linear, Huber)} \end{cases}$$

아웃라이어 factor의 영향을 saturation으로 제한.

MAP 최적화 — 수렴 분석

수렴 보장 조건

- H가 positive definite (관측 가능)
- 초기값이 수렴 반경 내
- Residual이 smooth (미분 가능)
- 충분한 iteration

실패 상황

- 잘못된 루프 클로저
- 초기값 불량 (global minimum 못 찾음)
- Degenerate motion (관측 불가 방향)
- 수치 오류 (scale 차이)

Robustness: Robust Cost Functions

Gaussian 대신 **Huber**, **Cauchy**, **GM** 등 robust loss:

$$\rho(r) = \begin{cases} \frac{1}{2}r^2 & |r| \le k \quad \text{(quadratic)} \\ k|r| - \frac{k^2}{2} & |r| > k \quad \text{(linear, Huber)} \end{cases}$$

아웃라이어 factor의 영향을 saturation으로 제한.

실용

GTSAM: `noiseModel::Robust::Create(HuberModel, baseNoise)`
Ceres: `new ceres::HuberLoss(k)` 또는 `CauchyLoss`

희소 행렬 연산 — 실용 비교

방법	복잡도	메모리	장점
Dense Cholesky	$O(n^3)$	$O(n^2)$	단순, 작은 시스템
Sparse Cholesky	$O(n \cdot k^2)$	$O(n \cdot k)$	chain graph
Schur complement	$O(m^3 + n \cdot m^2)$	$O(m^2)$	VSLAM
iSAM2	$O(1)$ amortized	$O(n)$	incremental

(n: poses, m: landmarks, k: bandwidth)

희소 행렬 연산 — 실용 비교

방법	복잡도	메모리	장점
Dense Cholesky	$O(n^3)$	$O(n^2)$	단순, 작은 시스템
Sparse Cholesky	$O(n \cdot k^2)$	$O(n \cdot k)$	chain graph
Schur complement	$O(m^3 + n \cdot m^2)$	$O(m^2)$	VSLAM
iSAM2	$O(1)$ amortized	$O(n)$	incremental

(n: poses, m: landmarks, k: bandwidth)

Variable Elimination Ordering

- Landmarks를 먼저 eliminate (Schur complement와 동일 효과)
- Poses를 나중에 (dense sub-problem)
- COLAMD가 자동으로 좋은 ordering 탐색

Ceres Solver — 범용 비선형 최소제곱

Google의 오픈소스 C++ nonlinear least squares library.

```
ceres::Problem problem;

// 각 residual block 추가
// ImuCostFunctor: operator() 에서 residual 계산
problem.AddResidualBlock(
    new ceres::AutoDiffCostFunction<ImuCostFunctor, 15, 6, 3, 6, 3, 6>(
        new ImuCostFunctor(preint_meas)),
    new ceres::HuberLoss(1.0),       // robust loss function
    pose_i, vel_i, pose_j, vel_j, bias);

// SO(3) manifold 설정
problem.SetManifold(pose_i, new ceres::EigenQuaternionManifold());

ceres::Solver::Options options;
options.linear_solver_type       = ceres::SPARSE_SCHUR;
options.trust_region_strategy_type = ceres::LEVENBERG_MARQUARDT;
options.num_threads              = 4;
ceres::Solve(options, &problem, &summary);
```

사용 프로젝트
COLMAP (SfM), Cartographer (LiDAR SLAM), Open3D, Kalibr (calibration)

g2o — Graph Optimization Library


```cpp
#include <g2o/core/sparse_optimizer.h>
#include <g2o/solvers/cholmod/linear_solver_cholmod.h>

g2o::SparseOptimizer optimizer;
// Levenberg-Marquardt + Cholmod solver 조합
auto linearSolver =
    std::make_unique<g2o::LinearSolverCholmod<g2o::BlockSolverX::PoseMatrixType>>();
auto blockSolver  = std::make_unique<g2o::BlockSolverX>(std::move(linearSolver));
auto algorithm    = new g2o::OptimizationAlgorithmLevenberg(std::move(blockSolver));
optimizer.setAlgorithm(algorithm);

// Vertex (pose, landmark) 추가
auto* v = new VertexSE3Expmap();
v->setId(id); v->setEstimate(pose); optimizer.addVertex(v);

// Edge (IMU, camera) 추가
auto* e = new EdgeSE3ProjectXYZ();
e->setVertex(0, optimizer.vertex(pose_id));
e->setVertex(1, optimizer.vertex(landmark_id));
e->setMeasurement(obs); e->setInformation(omega);
optimizer.addEdge(e);

optimizer.optimize(10);   // 10 LM iterations
```

사용 프로젝트
ORB-SLAM3, RTAB-Map, LIO-SAM (pose graph part)

Manifold 최적화 — SO(3)/SE(3) 위에서

Euclidean 공간 가정이 깨지는 곳: **회전**

$$R \in \mathrm{SO}(3) \subset \mathbb{R}^{3\times 3}, \quad \text{but optimizer는 } \mathbb{R}^n \text{에서 동작}$$

Manifold 최적화 — SO(3)/SE(3) 위에서

Euclidean 공간 가정이 깨지는 곳: **회전**

$$R \in \mathrm{SO}(3) \subset \mathbb{R}^{3\times 3}, \quad \text{but optimizer는 } \mathbb{R}^n \text{에서 동작}$$

Retraction $\boxplus$ and Local Coordinates

$$R \boxplus \delta\phi = R\,\mathrm{Exp}(\delta\phi), \qquad \delta\phi \in \mathbb{R}^3$$

Update는 Lie algebra $\mathfrak{so}(3)$에서, 결과는 $\mathrm{SO}(3)$로 map.

Manifold 최적화 — SO(3)/SE(3) 위에서

Euclidean 공간 가정이 깨지는 곳: **회전**

$$R \in \mathrm{SO}(3) \subset \mathbb{R}^{3\times 3}, \quad \text{but optimizer는 } \mathbb{R}^n \text{에서 동작}$$

Retraction $\boxplus$ and Local Coordinates

$$R \boxplus \delta\phi = R\,\mathrm{Exp}(\delta\phi), \qquad \delta\phi \in \mathbb{R}^3$$

Update는 Lie algebra $\mathfrak{so}(3)$에서, 결과는 $\mathrm{SO}(3)$로 map.

구현 방법:

- GTSAM: `Rot3`, `Pose3` 등이 모두 manifold 지원 내장
- Ceres: `EigenQuaternionManifold`, `SO3Manifold`
- g2o: `VertexSO3Expmap`

Manifold 최적화 — SO(3)/SE(3) 위에서

Euclidean 공간 가정이 깨지는 곳: **회전**

$$R \in \mathrm{SO}(3) \subset \mathbb{R}^{3\times 3}, \quad \text{but optimizer는 } \mathbb{R}^n \text{에서 동작}$$

Retraction $\boxplus$ and Local Coordinates

$$R \boxplus \delta\phi = R\,\mathrm{Exp}(\delta\phi), \qquad \delta\phi \in \mathbb{R}^3$$

Update는 Lie algebra $\mathfrak{so}(3)$에서, 결과는 $\mathrm{SO}(3)$로 map.

구현 방법:

- GTSAM: `Rot3`, `Pose3` 등이 모두 manifold 지원 내장
- Ceres: `EigenQuaternionManifold`, `SO3Manifold`
- g2o: `VertexSO3Expmap`

주의
Quaternion q로 parameterize 시 unit-norm constraint 필요.
Perturbation: $q \leftarrow q \otimes \mathrm{Exp}(\delta\phi/2)$ (half-angle)

Marginalization — 왜 필요한가?

Sliding Window Estimator의 딜레마
모든 과거 keyframe을 유지 $\rightarrow$ 메모리 $O(n^2)$, 계산 $O(n^3)$ 불가피
과거 keyframe을 단순 삭제 $\rightarrow$ 그 frame과 연결된 information 손실

Marginalization — 왜 필요한가?

Sliding Window Estimator의 딜레마
모든 과거 keyframe을 유지 $\rightarrow$ 메모리 $O(n^2)$, 계산 $O(n^3)$ 불가피
과거 keyframe을 단순 삭제 $\rightarrow$ 그 frame과 연결된 information 손실

문제 상황:

- Frame i가 landmarks l_1, l_2, l_3를 관측 후 윈도우에서 제거
- l_1, l_2, l_3는 이후에도 사용됨
- i를 그냥 지우면 l_1, l_2, l_3에 가해진 constraint가 사라짐

Marginalization — 왜 필요한가?

Sliding Window Estimator의 딜레마
모든 과거 keyframe을 유지 $\rightarrow$ 메모리 $O(n^2)$, 계산 $O(n^3)$ 불가피
과거 keyframe을 단순 삭제 $\rightarrow$ 그 frame과 연결된 information 손실

문제 상황:

- Frame i가 landmarks l_1, l_2, l_3를 관측 후 윈도우에서 제거
- l_1, l_2, l_3는 이후에도 사용됨
- i를 그냥 지우면 l_1, l_2, l_3에 가해진 constraint가 사라짐

해결책: Marginalization

- 제거할 변수의 information을 나머지 변수에 prior로 전달
- Gaussian이면 Schur complement로 **정확히** 수행 가능
- Information 손실 없이 변수 제거

Marginalization — 왜 필요한가?

Sliding Window Estimator의 딜레마
모든 과거 keyframe을 유지 $\rightarrow$ 메모리 $O(n^2)$, 계산 $O(n^3)$ 불가피
과거 keyframe을 단순 삭제 $\rightarrow$ 그 frame과 연결된 information 손실

문제 상황:

- Frame i가 landmarks l_1, l_2, l_3를 관측 후 윈도우에서 제거
- l_1, l_2, l_3는 이후에도 사용됨
- i를 그냥 지우면 l_1, l_2, l_3에 가해진 constraint가 사라짐

해결책: Marginalization

- 제거할 변수의 information을 나머지 변수에 prior로 전달
- Gaussian이면 Schur complement로 **정확히** 수행 가능
- Information 손실 없이 변수 제거

핵심 개념
Marginalization = 오래된 변수의 information을 “압축”해서 prior factor로 전환

Marginalization 이론 — Gaussian Case

Joint distribution:

$$p(x_a, x_b) = \mathcal{N}(\mu,\, \Sigma)\,, \qquad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$$

Marginalization 이론 — Gaussian Case

Joint distribution:

$$p(x_a, x_b) = \mathcal{N}(\mu,\, \Sigma)\,, \qquad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$$

Marginal (x_a 적분 제거):

$$p(x_b) = \int p(x_a, x_b)\, dx_a = \mathcal{N}(\mu_b,\, \Sigma_{bb})$$

Marginalization 이론 — Gaussian Case

Joint distribution:

$$p(x_a, x_b) = \mathcal{N}(\mu,\, \Sigma)\,, \qquad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$$

Marginal (x_a 적분 제거):

$$p(x_b) = \int p(x_a, x_b)\, dx_a = \mathcal{N}(\mu_b,\, \Sigma_{bb})$$

Conditional ($x_a = \hat{x}_a$ 조건부):

$$p(x_b \mid x_a) = \mathcal{N}\big(\mu_{b|a},\, \Sigma_{b|a}\big)$$
$$\Sigma_{b|a} = \Sigma_{bb} - \Sigma_{ba}\,\Sigma_{aa}^{-1}\,\Sigma_{ab} \quad \leftarrow \text{Schur complement!}$$

Marginalization 이론 — Gaussian Case

Joint distribution:

$$p(x_a, x_b) = \mathcal{N}(\mu,\, \Sigma)\,, \qquad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$$

Marginal (x_a 적분 제거):

$$p(x_b) = \int p(x_a, x_b)\, dx_a = \mathcal{N}(\mu_b,\, \Sigma_{bb})$$

Conditional ($x_a = \hat{x}_a$ 조건부):

$$p(x_b \mid x_a) = \mathcal{N}\big(\mu_{b|a},\, \Sigma_{b|a}\big)$$
$$\Sigma_{b|a} = \Sigma_{bb} - \Sigma_{ba}\,\Sigma_{aa}^{-1}\,\Sigma_{ab} \quad \leftarrow \text{Schur complement!}$$

직관
x_a를 고정 (또는 제거)하면 x_b의 불확실성은 줄어든다 ($\Sigma_{b|a} \le \Sigma_{bb}$).
이 줄어든 불확실성이 바로 x_a의 information이 x_b로 전달된 결과.

Information 행렬에서의 Marginalization

Information form (canonical form):

$$p(\boldsymbol{x}) \propto \exp\left(-\tfrac{1}{2}\boldsymbol{x}^\top \Omega\, \boldsymbol{x} + \xi^\top \boldsymbol{x}\right), \quad \Omega = \Sigma^{-1}, \quad \xi = \Omega\, \mu$$

분할:

$$\Omega = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}, \quad \xi = \begin{pmatrix} \xi_a \\ \xi_b \end{pmatrix}$$

Information 행렬에서의 Marginalization

Information form (canonical form):

$$p(\boldsymbol{x}) \propto \exp\left(-\tfrac{1}{2}\boldsymbol{x}^\top \Omega\, \boldsymbol{x} + \xi^\top \boldsymbol{x}\right), \quad \Omega = \Sigma^{-1}, \quad \xi = \Omega\, \mu$$

분할:

$$\Omega = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}, \quad \xi = \begin{pmatrix} \xi_a \\ \xi_b \end{pmatrix}$$

x_a를 marginalize out한 결과 — x_b의 prior

$$\Omega_b^{\text{marg}} = \Lambda_{bb} - \Lambda_{ba}\,\Lambda_{aa}^{-1}\,\Lambda_{ab}$$
$$\xi_b^{\text{marg}} = \xi_b - \Lambda_{ba}\,\Lambda_{aa}^{-1}\,\xi_a$$

Information 행렬에서의 Marginalization

Information form (canonical form):

$$p(\boldsymbol{x}) \propto \exp\left(-\tfrac{1}{2}\boldsymbol{x}^\top \Omega\, \boldsymbol{x} + \xi^\top \boldsymbol{x}\right), \quad \Omega = \Sigma^{-1}, \quad \xi = \Omega\, \mu$$

분할:

$$\Omega = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}, \quad \xi = \begin{pmatrix} \xi_a \\ \xi_b \end{pmatrix}$$

x_a를 marginalize out한 결과 — x_b의 prior

$$\Omega_b^{\text{marg}} = \Lambda_{bb} - \Lambda_{ba}\,\Lambda_{aa}^{-1}\,\Lambda_{ab}$$
$$\xi_b^{\text{marg}} = \xi_b - \Lambda_{ba}\,\Lambda_{aa}^{-1}\,\xi_a$$

핵심: Information form의 장점
Marginalization (information form) = **Schur complement** (covariance form)
Information 행렬에서 marginalization이 더 직접적으로 표현됨!

Information 행렬에서의 Marginalization

Information form (canonical form):

$$p(\boldsymbol{x}) \propto \exp\left(-\tfrac{1}{2}\boldsymbol{x}^\top \Omega\, \boldsymbol{x} + \xi^\top \boldsymbol{x}\right), \quad \Omega = \Sigma^{-1}, \quad \xi = \Omega\, \mu$$

분할:

$$\Omega = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}, \quad \xi = \begin{pmatrix} \xi_a \\ \xi_b \end{pmatrix}$$

x_a를 marginalize out한 결과 — x_b의 prior

$$\Omega_b^{\text{marg}} = \Lambda_{bb} - \Lambda_{ba}\,\Lambda_{aa}^{-1}\,\Lambda_{ab}$$
$$\xi_b^{\text{marg}} = \xi_b - \Lambda_{ba}\,\Lambda_{aa}^{-1}\,\xi_a$$

핵심: Information form의 장점
Marginalization (information form) = **Schur complement** (covariance form)
Information 행렬에서 marginalization이 더 직접적으로 표현됨!

결과 해석
$\Omega_b^{\text{marg.}}$: x_a가 x_b에 준 정보를 모두 흡수한 새 prior
Prior factor 형태: $\|x_b - x_b^*\|^2_{\Omega_b^{\text{marg}}}$

Sliding Window Marginalization — 단계별 과정

1. **Identify**: 제거할 변수 x_a 결정
   (oldest keyframe + 그와만 연결된 landmarks)

Sliding Window Marginalization — 단계별 과정

1. **Identify**: 제거할 변수 x_a 결정
   (oldest keyframe + 그와만 연결된 landmarks)
2. **Linearize**: x_a와 연결된 모든 factor 선형화
   $r_i(x_a, x_b) \approx r_i^0 + J_{ia}\,\delta x_a + J_{ib}\,\delta x_b$
   중요: 현재 linearization point에서 고정

Sliding Window Marginalization — 단계별 과정

1. **Identify**: 제거할 변수 x_a 결정
   (oldest keyframe + 그와만 연결된 landmarks)
2. **Linearize**: x_a와 연결된 모든 factor 선형화
   $r_i(x_a, x_b) \approx r_i^0 + J_{ia}\,\delta x_a + J_{ib}\,\delta x_b$
   중요: 현재 linearization point에서 고정
3. **Schur Complement**: x_a eliminate
   $\Omega_b^{\text{marg}}, \xi_b^{\text{marg}}$ 계산

Sliding Window Marginalization — 단계별 과정

1. **Identify**: 제거할 변수 x_a 결정
   (oldest keyframe + 그와만 연결된 landmarks)
2. **Linearize**: x_a와 연결된 모든 factor 선형화
   $r_i(x_a, x_b) \approx r_i^0 + J_{ia}\,\delta x_a + J_{ib}\,\delta x_b$
   중요: 현재 linearization point에서 고정
3. **Schur Complement**: x_a eliminate
   $\Omega_b^{\text{marg}}, \xi_b^{\text{marg}}$ 계산
4. **Prior Factor 생성**: 결과를 나머지 변수 x_b에 대한
   Gaussian prior로 추가: $\|x_b - x_b^*\|^2_{\Omega_b^{\text{marg}}}$

Sliding Window Marginalization — 단계별 과정

1. **Identify**: 제거할 변수 x_a 결정
   (oldest keyframe + 그와만 연결된 landmarks)
2. **Linearize**: x_a와 연결된 모든 factor 선형화
   $r_i(x_a, x_b) \approx r_i^0 + J_{ia}\,\delta x_a + J_{ib}\,\delta x_b$
   중요: 현재 linearization point에서 고정
3. **Schur Complement**: x_a eliminate
   $\Omega_b^{\text{marg}}, \xi_b^{\text{marg}}$ 계산
4. **Prior Factor 생성**: 결과를 나머지 변수 x_b에 대한
   Gaussian prior로 추가: $\|x_b - x_b^*\|^2_{\Omega_b^{\text{marg}}}$
5. **Remove**: x_a와 연결된 원래 factor들을 삭제,
   x_a 변수 제거 (prior factor만 남김)

Sliding Window Marginalization — 단계별 과정

1. **Identify**: 제거할 변수 x_a 결정
   (oldest keyframe + 그와만 연결된 landmarks)
2. **Linearize**: x_a와 연결된 모든 factor 선형화
   $r_i(x_a, x_b) \approx r_i^0 + J_{ia}\,\delta x_a + J_{ib}\,\delta x_b$
   중요: 현재 linearization point에서 고정
3. **Schur Complement**: x_a eliminate
   $\Omega_b^{\text{marg}}, \xi_b^{\text{marg}}$ 계산
4. **Prior Factor 생성**: 결과를 나머지 변수 x_b에 대한
   Gaussian prior로 추가: $\|x_b - x_b^*\|^2_{\Omega_b^{\text{marg}}}$
5. **Remove**: x_a와 연결된 원래 factor들을 삭제,
   x_a 변수 제거 (prior factor만 남김)

Linearization Point 고정 문제
Step 2에서 고정된 linearization point가 이후 추정값과 멀어지면
prior가 inconsistent해짐 → FEJ (First Estimates Jacobian)로 대응 (8회차)

Prior Factor as Marginalization Result

Marginalization 결과 = 비선형 prior factor의 선형 근사

$$f_{\text{prior}}(x_b) = \|x_b \ominus x_b^*\|^2_{\Omega^{\text{marg}}}$$

($\ominus$: manifold에서의 차이, $= \text{Log}(x_b^{*-1} \cdot x_b)$ on SE(3))

Prior Factor as Marginalization Result

Marginalization 결과 = 비선형 prior factor의 선형 근사

$$f_{\text{prior}}(x_b) = \|x_b \ominus x_b^*\|^2_{\Omega^{\text{marg}}}$$

($\ominus$: manifold에서의 차이, $= \text{Log}(x_b^{*-1} \cdot x_b)$ on SE(3))

특성

- 단 한 번의 선형화 근사
- 이후 x_b가 변해도 linearization point 고정
- Dense block: marginalize된 x_a가 연결했던 모든 x_b들 간에 correlation 생성
- Prior의 Ω^{marg}는 일반적으로 dense (full rank)

문제점

- 시간이 지나 x_b가 많이 변하면 prior inconsistent
- Spurious information 축적 가능
- 주기적 marginalization으로 관리 필요

Prior Factor as Marginalization Result

Marginalization 결과 = 비선형 prior factor의 선형 근사

$$f_{\text{prior}}(x_b) = \|x_b \ominus x_b^*\|^2_{\Omega^{\text{marg}}}$$

($\ominus$: manifold에서의 차이, $= \text{Log}(x_b^{*-1} \cdot x_b)$ on SE(3))

특성

- 단 한 번의 선형화 근사
- 이후 x_b가 변해도 linearization point 고정
- Dense block: marginalize된 x_a가 연결했던 모든 x_b들 간에 correlation 생성
- Prior의 Ω^{marg}는 일반적으로 dense (full rank)

문제점

- 시간이 지나 x_b가 많이 변하면 prior inconsistent
- Spurious information 축적 가능
- 주기적 marginalization으로 관리 필요

구현 예

OKVIS (Leutenegger et al., 2015): marginalization prior를 명시적으로 관리
Kimera-VIO: GTSAM의 `LinearContainerFactor`로 구현

Marginalization — 수식 유도 (전체)

x_a와 연결된 factor들로부터 local linear system:

$$\begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix} \begin{pmatrix} \delta x_a \\ \delta x_b \end{pmatrix} = - \begin{pmatrix} \xi_a \\ \xi_b \end{pmatrix}$$

Marginalization — 수식 유도 (전체)

x_a와 연결된 factor들로부터 local linear system:

$$\begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix} \begin{pmatrix} \delta x_a \\ \delta x_b \end{pmatrix} = - \begin{pmatrix} \xi_a \\ \xi_b \end{pmatrix}$$

Block elimination (x_a 제거):

$$\Lambda_{aa}\,\delta x_a = -\xi_a - \Lambda_{ab}\,\delta x_b$$

$$\delta x_a = -\Lambda_{aa}^{-1}(\xi_a + \Lambda_{ab}\,\delta x_b)$$

Marginalization — 수식 유도 (전체)

x_a와 연결된 factor들로부터 local linear system:

$$\begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix} \begin{pmatrix} \delta x_a \\ \delta x_b \end{pmatrix} = - \begin{pmatrix} \xi_a \\ \xi_b \end{pmatrix}$$

Block elimination (x_a 제거):

$$\Lambda_{aa}\,\delta x_a = -\xi_a - \Lambda_{ab}\,\delta x_b$$

$$\delta x_a = -\Lambda_{aa}^{-1}(\xi_a + \Lambda_{ab}\,\delta x_b)$$

δx_b 방정식에 대입:

$$\underbrace{(\Lambda_{bb} - \Lambda_{ba}\Lambda_{aa}^{-1}\Lambda_{ab})}_{\Omega^{\text{marg}}}\,\delta x_b = -\underbrace{(\xi_b - \Lambda_{ba}\Lambda_{aa}^{-1}\xi_a)}_{\xi^{\text{marg}}}$$

Marginalization — 수식 유도 (전체)

x_a와 연결된 factor들로부터 local linear system:

$$\begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix} \begin{pmatrix} \delta x_a \\ \delta x_b \end{pmatrix} = - \begin{pmatrix} \xi_a \\ \xi_b \end{pmatrix}$$

Block elimination (x_a 제거):

$$\Lambda_{aa}\,\delta x_a = -\xi_a - \Lambda_{ab}\,\delta x_b$$

$$\delta x_a = -\Lambda_{aa}^{-1}(\xi_a + \Lambda_{ab}\,\delta x_b)$$

δx_b 방정식에 대입:

$$\underbrace{(\Lambda_{bb} - \Lambda_{ba}\Lambda_{aa}^{-1}\Lambda_{ab})}_{\Omega^{\text{marg}}}\,\delta x_b = -\underbrace{(\xi_b - \Lambda_{ba}\Lambda_{aa}^{-1}\xi_a)}_{\xi^{\text{marg}}}$$

결과

$$\Omega^{\text{marg}} = \Lambda_{bb} - \Lambda_{ba}\Lambda_{aa}^{-1}\Lambda_{ab}, \qquad \xi^{\text{marg}} = \xi_b - \Lambda_{ba}\Lambda_{aa}^{-1}\xi_a$$

Fixed-lag Smoother vs. Full Bundle Adjustment

Full Bundle Adjustment (BA)

- 모든 keyframe을 한꺼번에 최적화
- 가장 정확 (global consistency)
- 메모리: $O(n^2)$
- 계산: $O(n^3)$ per iteration
- 오프라인 처리에 적합
- COLMAP, OpenSfM

Fixed-lag Smoother

- 최근 W개 keyframe만 윈도우 유지
- 메모리: $O(W^2)$ (bounded)
- 계산: $O(W^3)$ per step
- 오래된 것 $\rightarrow$ marginalization
- 실시간 가능
- VINS-Mono ($W = 10$), OKVIS

Fixed-lag Smoother vs. Full Bundle Adjustment

Full Bundle Adjustment (BA)

- 모든 keyframe을 한꺼번에 최적화
- 가장 정확 (global consistency)
- 메모리: $O(n^2)$
- 계산: $O(n^3)$ per iteration
- 오프라인 처리에 적합
- COLMAP, OpenSfM

Fixed-lag Smoother

- 최근 W개 keyframe만 윈도우 유지
- 메모리: $O(W^2)$ (bounded)
- 계산: $O(W^3)$ per step
- 오래된 것 $\rightarrow$ marginalization
- 실시간 가능
- VINS-Mono ($W = 10$), OKVIS

시스템	방법	Window	특징
VINS-Mono	Fixed-lag	10 KF	Marginalization
OKVIS	Fixed-lag	5 KF	ISAM2-style
Kimera-VIO	iSAM2	adaptive	Bayes tree
ORB-SLAM3	Full BA	local+global	두 단계
COLMAP	Full BA	all	오프라인 SfM

Filtering vs. Smoothing 재고찰

ESKF (Filtering)

- 순차적 처리, $O(1)$ per step
- Propagation + Update
- 선형화: 현재 추정값에서 1회
- 메모리: $O(n^2)$ (n: state 차원, 고정)
- 과거 추정값 수정 불가
- OpenVINS, FAST-LIO, LOAM

MAP Smoothing

- 윈도우 전체 반복 최적화
- Gauss-Newton / LM
- 선형화: 매 iteration마다 갱신
- 메모리: window 크기에 비례
- 윈도우 내 과거 추정값 수정 가능
- GTSAM/iSAM2, g2o-based

Filtering vs. Smoothing 재고찰

ESKF (Filtering)

- 순차적 처리, $O(1)$ per step
- Propagation + Update
- 선형화: 현재 추정값에서 1회
- 메모리: $O(n^2)$ (n: state 차원, 고정)
- 과거 추정값 수정 불가
- OpenVINS, FAST-LIO, LOAM

MAP Smoothing

- 윈도우 전체 반복 최적화
- Gauss-Newton / LM
- 선형화: 매 iteration마다 갱신
- 메모리: window 크기에 비례
- 윈도우 내 과거 추정값 수정 가능
- GTSAM/iSAM2, g2o-based

이론적 동등성 vs. 실제 차이

Gaussian, linear 가정 하에서는 **이론적으로 동등** (EKF = MAP with window=1)

실제: 비선형성, 여러 번의 재선형화, loop closure 처리에서 큰 차이 발생

Filtering vs. Smoothing 재고찰

ESKF (Filtering)

- 순차적 처리, $O(1)$ per step
- Propagation + Update
- 선형화: 현재 추정값에서 1회
- 메모리: $O(n^2)$ (n: state 차원, 고정)
- 과거 추정값 수정 불가
- OpenVINS, FAST-LIO, LOAM

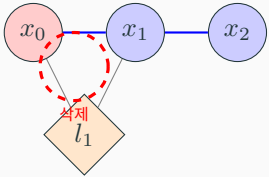
MAP Smoothing

- 윈도우 전체 반복 최적화
- Gauss-Newton / LM
- 선형화: 매 iteration마다 갱신
- 메모리: window 크기에 비례
- 윈도우 내 과거 추정값 수정 가능
- GTSAM/iSAM2, g2o-based

이론적 동등성 vs. 실제 차이
Gaussian, linear 가정 하에서는 **이론적으로 동등** (EKF = MAP with window=1)
실제: 비선형성, 여러 번의 재선형화, loop closure 처리에서 큰 차이 발생

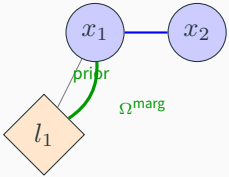
언제 무엇을?
ESKF: 실시간, 단순 센서 퓨전, 계산 제약 환경
MAP: 정밀도 중요, 루프 클로저 필요, 오프라인 가능

Marginalization vs. 단순 삭제 — 시뮬레이션 비교



단순 삭제 (Drop)

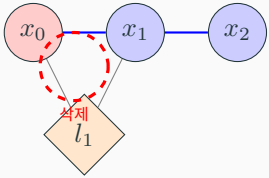
- $x_0 \to l_1$ constraint **소실**
- l_1의 불확실성 증가
- Inconsistent estimator



Marginalization

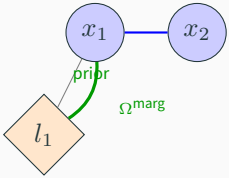
- x_0의 정보가 prior로 전달
- l_1, x_1 간 correlation 유지
- **Consistent** estimator

Marginalization vs. 단순 삭제 — 시뮬레이션 비교



단순 삭제 (Drop)

- $x_0 \to l_1$ constraint **소실**
- l_1의 불확실성 증가
- Inconsistent estimator



Marginalization

- x_0의 정보가 prior로 전달
- l_1, x_1 간 correlation 유지
- **Consistent** estimator

수치 예시

x_0 삭제 시 l_1의 covariance: Drop $\to 2\sigma^2$, Marginalize $\to 0.5\sigma^2$

Marginalization — 연결 변수 선택 전략

어떤 변수를 marginalize할 것인가?

1. **Oldest keyframe**: 시간 순서로 가장 오래된 frame
2. **Redundant keyframe**: 가장 최근 frame과 motion이 작은 frame
3. **Associated features**: marginalize되는 frame에만 보이는 landmarks

Marginalization — 연결 변수 선택 전략

어떤 변수를 marginalize할 것인가?

1. **Oldest keyframe**: 시간 순서로 가장 오래된 frame
2. **Redundant keyframe**: 가장 최근 frame과 motion이 작은 frame
3. **Associated features**: marginalize되는 frame에만 보이는 landmarks

중요한 규칙: marginalize할 변수는 남아 있는 변수들과의 연결이 최소여야 함

- 연결이 많으면 Ω^{marg}가 dense해져 prior 처리가 무거워짐
- Sparsification: 약한 연결은 무시 (Drop edge with small information)

Marginalization — 연결 변수 선택 전략

어떤 변수를 marginalize할 것인가?

1. **Oldest keyframe**: 시간 순서로 가장 오래된 frame
2. **Redundant keyframe**: 가장 최근 frame과 motion이 작은 frame
3. **Associated features**: marginalize되는 frame에만 보이는 landmarks

중요한 규칙: marginalize할 변수는 남아 있는 변수들과의 연결이 최소여야 함

- 연결이 많으면 Ω^{marg}가 dense해져 prior 처리가 무거워짐
- Sparsification: 약한 연결은 무시 (Drop edge with small information)

VINS-Mono 전략

- 충분한 시차(parallax) 없으면 second-newest frame marginalize
- 충분한 시차 있으면 oldest frame marginalize
- 목적: 윈도우 내 정보량 최대화

OKVIS 전략

- 5 keyframes + 5 temporal frames
- Keyframe 제거 시 marginalization
- Temporal frame 제거 시 drop

FEJ (First Estimate Jacobian) — Marginalization과의 연결

문제: Marginalization에서 linearization point를 한번 고정

- Prior factor의 Jacobian은 $\hat{x}_b^{(\text{fixed})}$에서 계산됨
- 이후 optimization에서 x_b의 estimate가 바뀌어도 Jacobian 고정
- Prior factor와 새 factor의 linearization point가 불일치 $\rightarrow$ inconsistency

FEJ (First Estimate Jacobian) — Marginalization과의 연결

문제: Marginalization에서 linearization point를 한번 고정

- Prior factor의 Jacobian은 $\hat{x}_b^{(\text{fixed})}$에서 계산됨
- 이후 optimization에서 x_b의 estimate가 바뀌어도 Jacobian 고정
- Prior factor와 새 factor의 linearization point가 불일치 $\rightarrow$ inconsistency

FEJ (First Estimate Jacobian) 해결책
모든 factor의 Jacobian을 처음 linearize된 시점의 값으로 고정
$\Rightarrow$ marginalization의 linearization point와 일관성 유지

FEJ (First Estimate Jacobian) — Marginalization과의 연결

문제: Marginalization에서 linearization point를 한번 고정

- Prior factor의 Jacobian은 $\hat{x}_b^{(\text{fixed})}$에서 계산됨
- 이후 optimization에서 x_b의 estimate가 바뀌어도 Jacobian 고정
- Prior factor와 새 factor의 linearization point가 불일치 $\rightarrow$ inconsistency

FEJ (First Estimate Jacobian) 해결책
모든 factor의 Jacobian을 처음 linearize된 시점의 값으로 고정
$\Rightarrow$ marginalization의 linearization point와 일관성 유지

$$J_i^{\text{FEJ}} = \left.\frac{\partial r_i}{\partial x}\right|_{x=\hat{x}^{(1\text{st})}} \quad \text{(이후 업데이트에서도 동일 값 사용)}$$

FEJ (First Estimate Jacobian) — Marginalization과의 연결

문제: Marginalization에서 linearization point를 한번 고정

- Prior factor의 Jacobian은 $\hat{x}_b^{(\text{fixed})}$에서 계산됨
- 이후 optimization에서 x_b의 estimate가 바뀌어도 Jacobian 고정
- Prior factor와 새 factor의 linearization point가 불일치 $\rightarrow$ inconsistency

FEJ (First Estimate Jacobian) 해결책
모든 factor의 Jacobian을 처음 linearize된 시점의 값으로 고정
$\Rightarrow$ marginalization의 linearization point와 일관성 유지

$$J_i^{\text{FEJ}} = \left.\frac{\partial r_i}{\partial x}\right|_{x=\hat{x}^{(1\text{st})}} \quad \text{(이후 업데이트에서도 동일 값 사용)}$$

적용 시스템
MSCKF + FEJ: OpenVINS에서 관측가능성 보존에 활용
VINS-Fusion: marginalization과 FEJ 결합
이론: Huang et al., “Observability-based Rules for Designing Consistent EKF SLAM” (2010)

GTSAM — Marginalization 구현


```cpp
#include <gtsam/nonlinear/ISAM2.h>
#include <gtsam/nonlinear/LinearContainerFactor.h>

ISAM2 isam2;
NonlinearFactorGraph newFactors;
Values newValues;

// ... factor/value 추가후  ...
isam2.update(newFactors, newValues);

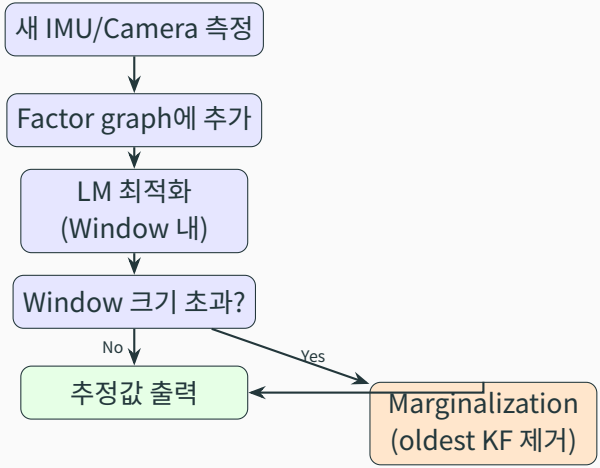
// 변수 marginalize (oldest keyframe 제거)
FastList<Key> marginalizeKeys;
marginalizeKeys.push_back(Symbol('x', oldest_id));
marginalizeKeys.push_back(Symbol('v', oldest_id));
marginalizeKeys.push_back(Symbol('b', oldest_id));

// Marginal 를factor 에graph 남김
ISAM2Result result = isam2.update();
// 는ISAM2 내부적으로 Bayes 를tree 통해 marginalization 처리
auto marginalFactor = isam2.marginalizeLeaves(marginalizeKeys);
```

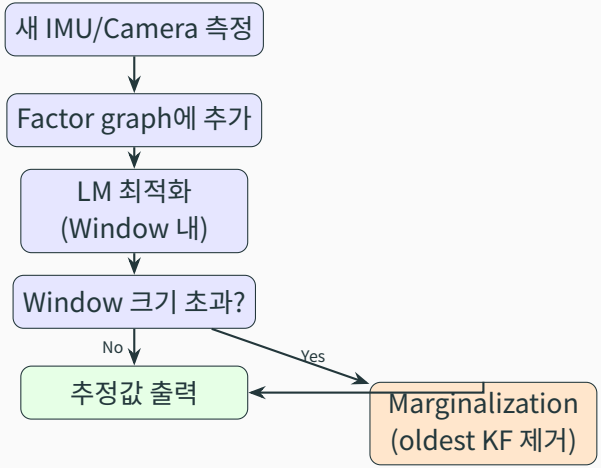
GTSAM의 marginalization 방식

marginalizeLeaves(): Bayes tree의 leaf 변수를 제거하고
해당 정보를 parent clique의 factor로 변환

Bounded Memory Optimization — 요약



Bounded Memory Optimization — 요약



메모리/계산 복잡도 보장

Window W로 고정 $\rightarrow$ 메모리 $O(W)$, 계산 $O(W^3)$ (bounded)

실제 VIO 시스템에서의 Marginalization

VINS-Mono (Hong Kong UST, 2018)

- Window: 10 keyframes
- Marginalization: Schur complement 직접 구현
- Feature: co-visibility graph로 keyframe 선택
- 논문: Qin et al., TRO 2018

Kimera-VIO (MIT SPARK, 2020)

- iSAM2 기반 (GTSAM)
- Bayes tree로 자동 marginalization
- Mesh 생성 + semantic 연동
- 논문: Rosinol et al., ICRA 2020

실제 VIO 시스템에서의 Marginalization

VINS-Mono (Hong Kong UST, 2018)

- Window: 10 keyframes
- Marginalization: Schur complement 직접 구현
- Feature: co-visibility graph로 keyframe 선택
- 논문: Qin et al., TRO 2018

OKVIS (ETH Zurich, 2015)

- Sliding window, Ceres Solver
- Marginalization prior 명시적 관리
- 4-cam stereo+IMU 지원
- 논문: Leutenegger et al., IJRR 2015

Kimera-VIO (MIT SPARK, 2020)

- iSAM2 기반 (GTSAM)
- Bayes tree로 자동 marginalization
- Mesh 생성 + semantic 연동
- 논문: Rosinol et al., ICRA 2020

OpenVINS (UofDelaware, 2020)

- MSCKF 기반 EKF (marginalization 없음)
- SLAM features로 부분 최적화
- 오픈소스, ROS2 지원
- 논문: Geneva et al., ICRA 2020

Loop Closure — MAP 최적화와의 결합

Loop Closure란?
이전에 방문한 장소를 다시 인식하여 누적 drift를 보정하는 것

Loop Closure — MAP 최적화와의 결합

Loop Closure란?
이전에 방문한 장소를 다시 인식하여 누적 drift를 보정하는 것

처리 방법 (MAP smoothing 관점):

1. Place recognition: bag-of-words (DBoW2, VLAD) 또는 학습 기반
2. Relative pose 추정: PnP, essential matrix
3. Loop closure factor 추가: $\|x_i \ominus x_j - \Delta T_{ij}\|^2_\Sigma$
4. Full BA 또는 pose graph optimization으로 global consistency

Loop Closure — MAP 최적화와의 결합

Loop Closure란?
이전에 방문한 장소를 다시 인식하여 누적 drift를 보정하는 것

처리 방법 (MAP smoothing 관점):

1. Place recognition: bag-of-words (DBoW2, VLAD) 또는 학습 기반
2. Relative pose 추정: PnP, essential matrix
3. Loop closure factor 추가: $\|x_i \ominus x_j - \Delta T_{ij}\|^2_\Sigma$
4. Full BA 또는 pose graph optimization으로 global consistency

Filtering에서의 어려움
ESKF는 과거 추정값을 수정할 수 없어 loop closure 처리가 매우 어려움
→ 별도 pose graph layer 필요 (SLAM 프레임워크의 “back-end”)

Loop Closure — MAP 최적화와의 결합

Loop Closure란?
이전에 방문한 장소를 다시 인식하여 누적 drift를 보정하는 것

처리 방법 (MAP smoothing 관점):

1. Place recognition: bag-of-words (DBoW2, VLAD) 또는 학습 기반
2. Relative pose 추정: PnP, essential matrix
3. Loop closure factor 추가: $\|x_i \ominus x_j - \Delta T_{ij}\|^2_\Sigma$
4. Full BA 또는 pose graph optimization으로 global consistency

Filtering에서의 어려움
ESKF는 과거 추정값을 수정할 수 없어 loop closure 처리가 매우 어려움
→ 별도 pose graph layer 필요 (SLAM 프레임워크의 “back-end”)

ORB-SLAM3의 접근
Front-end: feature tracking (EKF-like)
Back-end: local BA (g2o) + loop detection
Loop closure: essential graph optimization (pose graph)

Pose Graph Optimization — 경량화 BA

아이디어: feature를 모두 명시적으로 유지하지 않고, poses 간 relative constraint만 유지

$$F_{\text{pg}} = \sum_{(i,j)\in\mathcal{E}} \left\| \text{Log}\left(T_j^{-1}\, T_{ij}\, T_i \right) \right\|^2_{\Sigma_{ij}}$$

Pose Graph Optimization — 경량화 BA

아이디어: feature를 모두 명시적으로 유지하지 않고, poses 간 relative constraint만 유지

$$F_{\text{pg}} = \sum_{(i,j)\in\mathcal{E}} \left\| \text{Log}\left(T_j^{-1}\, T_{ij}\, T_i \right) \right\|^2_{\Sigma_{ij}}$$

장점

- 변수: poses만 (n개 $\times$ 6-DoF)
- 빠른 optimization
- Loop closure에 최적
- 전체 trajectory 일관성

단점 / 한계

- Feature position 정보 손실
- Relative pose uncertainty 누적
- Full BA보다 덜 정확
- Front-end에 의존

Pose Graph Optimization — 경량화 BA

아이디어: feature를 모두 명시적으로 유지하지 않고, poses 간 relative constraint만 유지

$$F_{\text{pg}} = \sum_{(i,j)\in\mathcal{E}} \left\| \text{Log}\left(T_j^{-1}\, T_{ij}\, T_i \right) \right\|^2_{\Sigma_{ij}}$$

장점

- 변수: poses만 (n개 $\times$ 6-DoF)
- 빠른 optimization
- Loop closure에 최적
- 전체 trajectory 일관성

단점 / 한계

- Feature position 정보 손실
- Relative pose uncertainty 누적
- Full BA보다 덜 정확
- Front-end에 의존

Edge 종류

- **Odometry edge**: 연속 pose 간 (IMU preintegration, wheel)
- **Loop closure edge**: 재방문 감지 시
- **GPS factor**: absolute position constraint

Pose Graph Optimization — 경량화 BA

아이디어: feature를 모두 명시적으로 유지하지 않고, poses 간 relative constraint만 유지

$$F_{\text{pg}} = \sum_{(i,j)\in\mathcal{E}} \left\| \text{Log}\left(T_j^{-1}\, T_{ij}\, T_i \right) \right\|^2_{\Sigma_{ij}}$$

장점

- 변수: poses만 (n개 $\times$ 6-DoF)
- 빠른 optimization
- Loop closure에 최적
- 전체 trajectory 일관성

단점 / 한계

- Feature position 정보 손실
- Relative pose uncertainty 누적
- Full BA보다 덜 정확
- Front-end에 의존

Edge 종류

- **Odometry edge**: 연속 pose 간 (IMU preintegration, wheel)
- **Loop closure edge**: 재방문 감지 시
- **GPS factor**: absolute position constraint

구현
g2o: `EdgeSE3` + `VertexSE3`
GTSAM: `BetweenFactor<Pose3>` + `NonlinearFactorGraph`

MAP 최적화 — 계산 비용 실전 분석

알고리즘	변수 수	Factor 수	시간/step	비고
Pose graph (g2o)	n poses	$\sim 2n$	$<$5ms	loop closure
Local BA (GTSAM)	n+m	$\sim 5n$	10–30ms	VIO window
Full BA (Ceres)	모두	모두	$>$1s	오프라인
iSAM2 increment	+1 KF	+few	$<$5ms	실시간

(n: poses, m: landmarks, 10–100 KF 기준 측정값)

MAP 최적화 — 계산 비용 실전 분석

알고리즘	변수 수	Factor 수	시간/step	비고
Pose graph (g2o)	n poses	$\sim 2n$	$<$5ms	loop closure
Local BA (GTSAM)	n+m	$\sim 5n$	10–30ms	VIO window
Full BA (Ceres)	모두	모두	$>$1s	오프라인
iSAM2 increment	+1 KF	+few	$<$5ms	실시간

(n: poses, m: landmarks, 10–100 KF 기준 측정값)

실시간 VIO 타이밍 (예: Intel i7, 30Hz camera)

- Feature tracking: 2–5ms
- IMU preintegration: $<$1ms
- MAP optimization (window=10): 10–20ms
- Total budget at 30Hz: 33ms

대안적 관점: MAP optimization을 message passing으로 해석

Sum-Product Algorithm on Factor Graph

$$\mu_{f_i \to x_j}(x_j) = \int \prod_{k \neq j} \mu_{x_k \to f_i}(x_k) \cdot f_i(x_j, \{x_k\})\, dx_k$$

Gaussian일 때: message = Gaussian, 곱셈 = information matrix 덧셈

대안적 관점: MAP optimization을 message passing으로 해석

Sum-Product Algorithm on Factor Graph

$$\mu_{f_i \to x_j}(x_j) = \int \prod_{k \neq j} \mu_{x_k \to f_i}(x_k) \cdot f_i(x_j, \{x_k\})\, dx_k$$

Gaussian일 때: message = Gaussian, 곱셈 = information matrix 덧셈

- Tree 구조: exact inference, $O(n)$
- Loopy graph: iterative (belief propagation, convergence 미보장)
- iSAM2의 Bayes tree = message passing의 효율적 구현

대안적 관점: MAP optimization을 message passing으로 해석

Sum-Product Algorithm on Factor Graph

$$\mu_{f_i \to x_j}(x_j) = \int \prod_{k \neq j} \mu_{x_k \to f_i}(x_k) \cdot f_i(x_j, \{x_k\})\, dx_k$$

Gaussian일 때: message = Gaussian, 곱셈 = information matrix 덧셈

- Tree 구조: exact inference, $O(n)$
- Loopy graph: iterative (belief propagation, convergence 미보장)
- iSAM2의 Bayes tree = message passing의 효율적 구현

연결
Kalman Filter: factor graph에서 chain topology의 message passing
RTS Smoother: forward (KF) + backward (message) pass
MAP optimization: iterative linearization + sparse linear solve

전체 VIO 파이프라인 — 코드 구조 (GTSAM 기반)


```
class VIOEstimator {
    ISAM2 isam2_;
    NonlinearFactorGraph newFactors_;
    Values newValues_;
    int current_kf_id_ = 0;

    void addImuFactor(int i, int j, PreintegratedCombinedMeasurements& pim) {
        newFactors_.add(CombinedImuFactor(X(i),V(i),X(j),V(j),B(i),B(j), pim));
    }
    void addCameraFactor(int kf_id, int lm_id, Vector2 obs, Cal3_S2 K) {
        newFactors_.add(GenericProjectionFactor<Pose3,Point3,Cal3_S2>(
            obs, noiseModel::Isotropic::Sigma(2, 1.5),
            X(kf_id), L(lm_id), boost::make_shared<Cal3_S2>(K)));
    }
    void marginalize(int oldest_id) {
        FastList<Key> toMarg = {X(oldest_id), V(oldest_id), B(oldest_id)};
        isam2_.marginalizeLeaves(toMarg);
    }
    Values update() {
        isam2_.update(newFactors_, newValues_);
        newFactors_.resize(0); newValues_.clear();
        return isam2_.calculateEstimate();
    }
};
```

Observability와 MAP 최적화의 관계

7회차 앞부분에서 배운 Observability를 MAP 관점에서 재해석

Observability와 MAP 최적화의 관계

7회차 앞부분에서 배운 Observability를 MAP 관점에서 재해석

Observability $\leftrightarrow$ Information Matrix

$$\text{Observable} \iff H = \sum_i J_i^\top \Sigma_i^{-1} J_i \text{ 가 positive definite}$$

Null space of H = Unobservable directions
H의 eigenvalue $\approx 0 \rightarrow$ 해당 방향 추정 불가 (발산)

Observability와 MAP 최적화의 관계

7회차 앞부분에서 배운 Observability를 MAP 관점에서 재해석

Observability ↔ Information Matrix

$$\text{Observable} \iff H = \sum_i J_i^\top \Sigma_i^{-1} J_i \text{ 가 positive definite}$$

Null space of H = Unobservable directions
H의 eigenvalue $\approx 0 \rightarrow$ 해당 방향 추정 불가 (발산)

SLAM의 대표적 unobservable 방향:

- **절대 위치**: 세계 좌표계 원점 (no GPS → unobservable)
- **절대 heading** (monocular+IMU): roll, pitch = observable, yaw = unobservable
- **절대 scale** (monocular): feature depth unobservable

Observability와 MAP 최적화의 관계

7회차 앞부분에서 배운 Observability를 MAP 관점에서 재해석

Observability ↔ Information Matrix

$$\text{Observable} \iff H = \sum_i J_i^\top \Sigma_i^{-1} J_i \text{ 가 positive definite}$$

Null space of H = Unobservable directions
H의 eigenvalue $\approx 0 \rightarrow$ 해당 방향 추정 불가 (발산)

SLAM의 대표적 unobservable 방향:

- **절대 위치**: 세계 좌표계 원점 (no GPS → unobservable)
- **절대 heading** (monocular+IMU): roll, pitch = observable, yaw = unobservable
- **절대 scale** (monocular): feature depth unobservable

MAP 최적화에서의 처리
Prior factor를 추가하여 null space를 fix:

- Position prior: x_0에 고정 prior
- Heading prior: 초기 orientation prior
- Scale: stereo 또는 depth sensor로 관측 가능하게

MAP 최적화 — 수치 안정성 기법

변수 정규화 (Normalization)

- Position: meter 단위 유지
- Rotation: unit quaternion enforce
- Landmark: 관측 가능한 범위로 초기화
- 센서 noise를 같은 스케일로 설정

Gauge Fixing

- 첫 pose를 world frame에 fix
- 또는 prior factor로 soft fix
- 절대 좌표계 고정 $\rightarrow$ H positive definite

Huber Loss (이상값 억제)

$$\rho_H(r) = \begin{cases} \frac{1}{2}r^2 & |r| \leq k \\ k|r| - \frac{k^2}{2} & |r| > k \end{cases}$$

Covariance Conditioning

- H에 작은 대각 damping: $H + \epsilon I$
- Condition number 모니터링
- 수치 Cholesky 실패 시 fallback

Iteration 제한

- Max iterations 설정
- Convergence 미달 시 경고

LiDAR 특화 MAP 최적화 기법:

LiDAR 특화 MAP 최적화 기법:

Scan Matching = 비선형 최소제곱

$$T^* = \arg\min_{T \in \mathrm{SE}(3)} \sum_i \| n_i^\top (T\, p_i - q_i) \|^2$$

p_i: source point, q_i: target point/plane, n_i: surface normal

LiDAR 특화 MAP 최적화 기법:

Scan Matching = 비선형 최소제곱

$$T^* = \arg\min_{T \in \mathrm{SE}(3)} \sum_i \| n_i^\top (T\, p_i - q_i) \|^2$$

p_i: source point, q_i: target point/plane, n_i: surface normal

대표 알고리즘:

- **ICP** (Iterative Closest Point): point-to-point, GN으로 수렴
- **NDT** (Normal Distribution Transform): voxel 단위 Gaussian
- **LOAM/LeGO-LOAM**: edge+surface feature matching, g2o
- **LIO-SAM**: IMU tight-coupled, GTSAM factor graph
- **FAST-LIO2**: iEKF (iterative EKF) + ikd-tree

LiDAR 특화 MAP 최적화 기법:

Scan Matching = 비선형 최소제곱

$$T^* = \arg\min_{T \in \mathrm{SE}(3)} \sum_i \| n_i^\top (T\, p_i - q_i) \|^2$$

p_i: source point, q_i: target point/plane, n_i: surface normal

대표 알고리즘:

- **ICP** (Iterative Closest Point): point-to-point, GN으로 수렴
- **NDT** (Normal Distribution Transform): voxel 단위 Gaussian
- **LOAM/LeGO-LOAM**: edge+surface feature matching, g2o
- **LIO-SAM**: IMU tight-coupled, GTSAM factor graph
- **FAST-LIO2**: iEKF (iterative EKF) + ikd-tree

LIO-SAM의 factor graph 구성
IMU preintegration factor + scan matching between factor + GPS factor + loop closure factor
GTSAM 기반, fixed-lag smoothing 적용

딥러닝과 MAP 최적화의 결합

학습 기반 front-end + MAP 기반 back-end 구조:

딥러닝과 MAP 최적화의 결합

학습 기반 front-end + MAP 기반 back-end 구조:

Front-end (학습 기반)

- Feature detection: SuperPoint, ALIKE
- Feature matching: SuperGlue, LightGlue
- Depth 추정: monodepth2, DPT
- Place recognition: NetVLAD, DBoW3
- Pose regression: PoseNet (참고용)

Back-end (MAP 최적화)

- GTSAM / g2o / Ceres
- Factor graph 구조 동일
- 학습 기반 noise model
- Uncertainty 정량화 필요

딥러닝과 MAP 최적화의 결합

학습 기반 front-end + MAP 기반 back-end 구조:

Front-end (학습 기반)

- Feature detection: SuperPoint, ALIKE
- Feature matching: SuperGlue, LightGlue
- Depth 추정: monodepth2, DPT
- Place recognition: NetVLAD, DBoW3
- Pose regression: PoseNet (참고용)

최신 트렌드
End-to-end differentiable SLAM: factor graph 내 학습 파라미터
DROID-SLAM: dense bundle adjustment + DNN, GRU로 update step 학습
DeepFactors: learned depth uncertainty를 factor graph에 직접 통합

Back-end (MAP 최적화)

- GTSAM / g2o / Ceres
- Factor graph 구조 동일
- 학습 기반 noise model
- Uncertainty 정량화 필요

딥러닝과 MAP 최적화의 결합

학습 기반 front-end + MAP 기반 back-end 구조:

Front-end (학습 기반)

- Feature detection: SuperPoint, ALIKE
- Feature matching: SuperGlue, LightGlue
- Depth 추정: monodepth2, DPT
- Place recognition: NetVLAD, DBoW3
- Pose regression: PoseNet (참고용)

Back-end (MAP 최적화)

- GTSAM / g2o / Ceres
- Factor graph 구조 동일
- 학습 기반 noise model
- Uncertainty 정량화 필요

최신 트렌드
End-to-end differentiable SLAM: factor graph 내 학습 파라미터
DROID-SLAM: dense bundle adjustment + DNN, GRU로 update step 학습
DeepFactors: learned depth uncertainty를 factor graph에 직접 통합

한계
학습 모델의 불확실성 추정이 부정확하면 MAP 최적화도 망가짐
⇒ calibration된 uncertainty (conformal prediction 등) 필요

요약: MAP 최적화 체계

1. **MAP = 비선형 최소제곱**
   $x^* = \arg\min \sum_i \|r_i(x)\|^2_{\Sigma_i}$, Gaussian noise $\rightarrow$ negative log-likelihood

요약: MAP 최적화 체계

1. **MAP = 비선형 최소제곱**
   $x^* = \arg\min \sum_i \|r_i(x)\|^2_{\Sigma_i}$, Gaussian noise $\rightarrow$ negative log-likelihood
2. **Gauss-Newton / Levenberg-Marquardt**
   Normal equation $H\,\delta x = -b$, LM damping으로 안정화
   manifold 위에서 retraction ⊞ 사용

요약: MAP 최적화 체계

1. **MAP = 비선형 최소제곱**
   $\boldsymbol{x}^* = \arg\min \sum_i \|r_i(\boldsymbol{x})\|^2_{\Sigma_i}$, Gaussian noise $\rightarrow$ negative log-likelihood
2. **Gauss-Newton / Levenberg-Marquardt**
   Normal equation $H\,\delta\boldsymbol{x} = -b$, LM damping으로 안정화
   manifold 위에서 retraction $\boxplus$ 사용
3. **SLAM의 희소성 활용**
   Sparse Cholesky, Schur complement, variable ordering (COLAMD)
   dense $O(n^3)$ $\rightarrow$ sparse $O(n)$ (chain graph)

요약: MAP 최적화 체계

1. **MAP = 비선형 최소제곱**
   $x^* = \arg\min \sum_i \|r_i(x)\|^2_{\Sigma_i}$, Gaussian noise $\rightarrow$ negative log-likelihood
2. **Gauss-Newton / Levenberg-Marquardt**
   Normal equation $H\,\delta x = -b$, LM damping으로 안정화
   manifold 위에서 retraction $\boxplus$ 사용
3. **SLAM의 희소성 활용**
   Sparse Cholesky, Schur complement, variable ordering (COLAMD)
   dense $O(n^3)$ $\rightarrow$ sparse $O(n)$ (chain graph)
4. **iSAM2: incremental 인수분해**
   Bayes tree + Wildfire: 영향받는 변수만 재계산
   단일 factor 추가 $\approx O(1)$ 실용적 복잡도

요약: MAP 최적화 체계

1. **MAP = 비선형 최소제곱**
   $\boldsymbol{x}^* = \arg\min \sum_i \|r_i(\boldsymbol{x})\|^2_{\Sigma_i}$, Gaussian noise $\rightarrow$ negative log-likelihood
2. **Gauss-Newton / Levenberg-Marquardt**
   Normal equation $H\,\delta\boldsymbol{x} = -b$, LM damping으로 안정화
   manifold 위에서 retraction $\boxplus$ 사용
3. **SLAM의 희소성 활용**
   Sparse Cholesky, Schur complement, variable ordering (COLAMD)
   dense $O(n^3)$ $\rightarrow$ sparse $O(n)$ (chain graph)
4. **iSAM2: incremental 인수분해**
   Bayes tree + Wildfire: 영향받는 변수만 재계산
   단일 factor 추가 $\approx O(1)$ 실용적 복잡도
5. **Marginalization: Schur complement로 오래된 변수 제거**
   information 보존하며 변수 제거, prior factor로 전달
   Fixed-lag smoother = bounded memory MAP optimization

요약: MAP 최적화 체계

1. **MAP = 비선형 최소제곱**
   $x^* = \arg\min \sum_i \|r_i(x)\|^2_{\Sigma_i}$, Gaussian noise → negative log-likelihood
2. **Gauss-Newton / Levenberg-Marquardt**
   Normal equation $H\,\delta x = -b$, LM damping으로 안정화
   manifold 위에서 retraction ⊞ 사용
3. **SLAM의 희소성 활용**
   Sparse Cholesky, Schur complement, variable ordering (COLAMD)
   dense $O(n^3)$ → sparse $O(n)$ (chain graph)
4. **iSAM2: incremental 인수분해**
   Bayes tree + Wildfire: 영향받는 변수만 재계산
   단일 factor 추가 ≈ $O(1)$ 실용적 복잡도
5. **Marginalization: Schur complement로 오래된 변수 제거**
   information 보존하며 변수 제거, prior factor로 전달
   Fixed-lag smoother = bounded memory MAP optimization

다음 회차 (L8) 예고
이 MAP 최적화가 **MSCKF**와 어떻게 다른가? FEJ, Nullspace projection,
관측가능성 보존 — EKF와 MAP의 이론적 통합 관점

Kalman Filter (Covariance form):

Predict: $\boldsymbol{P}^{-} = \boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^{\top} + \boldsymbol{Q}$

Update: $\boldsymbol{K} = \boldsymbol{P}^{-}\boldsymbol{H}^{\top}(\boldsymbol{H}\boldsymbol{P}^{-}\boldsymbol{H}^{\top} + \boldsymbol{R})^{-1}, \quad \boldsymbol{P}^{+} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}^{-}$

부록: Information Filter vs. Kalman Filter

Kalman Filter (Covariance form):

Predict: $\boldsymbol{P}^{-} = \boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^{\top} + \boldsymbol{Q}$

Update: $\boldsymbol{K} = \boldsymbol{P}^{-}\boldsymbol{H}^{\top}(\boldsymbol{H}\boldsymbol{P}^{-}\boldsymbol{H}^{\top} + \boldsymbol{R})^{-1}, \quad \boldsymbol{P}^{+} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}^{-}$

Information Filter ($\Omega = \boldsymbol{P}^{-1}, \xi = \Omega\,\hat{\boldsymbol{x}}$):

Predict: $\Omega^{-} = (\boldsymbol{F}\Omega^{-1}\boldsymbol{F}^{\top} + \boldsymbol{Q})^{-1}$

Update: $\Omega^{+} = \Omega^{-} + \boldsymbol{H}^{\top}\boldsymbol{R}^{-1}\boldsymbol{H}, \quad \xi^{+} = \xi^{-} + \boldsymbol{H}^{\top}\boldsymbol{R}^{-1}\boldsymbol{z}$

부록: Information Filter vs. Kalman Filter

Kalman Filter (Covariance form):

$$\text{Predict: } \boldsymbol{P}^{-} = \boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^{\top} + \boldsymbol{Q}$$

$$\text{Update: } \boldsymbol{K} = \boldsymbol{P}^{-}\boldsymbol{H}^{\top}(\boldsymbol{H}\boldsymbol{P}^{-}\boldsymbol{H}^{\top} + \boldsymbol{R})^{-1}, \quad \boldsymbol{P}^{+} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}^{-}$$

Information Filter ($\Omega = \boldsymbol{P}^{-1}, \xi = \Omega\,\hat{\boldsymbol{x}}$):

$$\text{Predict: } \Omega^{-} = (\boldsymbol{F}\Omega^{-1}\boldsymbol{F}^{\top} + \boldsymbol{Q})^{-1}$$

$$\text{Update: } \Omega^{+} = \Omega^{-} + \boldsymbol{H}^{\top}\boldsymbol{R}^{-1}\boldsymbol{H}, \quad \xi^{+} = \xi^{-} + \boldsymbol{H}^{\top}\boldsymbol{R}^{-1}\boldsymbol{z}$$

비교

- KF update: $O(n^2 m)$ (m: 측정 차원)
- IF update: $O(n^2 m)$ 비슷하나 measurement fusion이 더 간단
- IF predict: $O(n^3)$ (역행렬 필요) — 단점
- 분산 fusion, multi-robot에서 IF가 유리 (information 덧셈)

부록: Bundle Adjustment — SfM에서의 MAP

Classical Bundle Adjustment (Triggs et al., 1999):

$$\min_{\{T_i\},\{X_j\}} \sum_{i,j} \rho\Big(\|x_{ij} - \pi(T_i, X_j)\|^2_{\Sigma}\Big)$$

T_i: camera poses, X_j: 3D landmark, $\pi(\cdot)$: projection, ρ: robust loss

부록: Bundle Adjustment — SfM에서의 MAP

Classical Bundle Adjustment (Triggs et al., 1999):

$$\min_{\{T_i\},\{X_j\}} \sum_{i,j} \rho\Big(\|x_{ij} - \pi(T_i, X_j)\|^2_{\Sigma}\Big)$$

T_i: camera poses, X_j: 3D landmark, $\pi(\cdot)$: projection, ρ: robust loss

구조

- 변수: $n \times 6$-DoF poses + $m \times 3$-DoF landmarks
- Jacobian: $2 \times (6+3)$ per reprojection
- H 구조: “arrow” (poses) + “diagonal” (landmarks)
- Schur complement로 landmark 먼저 eliminate

Classical Bundle Adjustment (Triggs et al., 1999):

$$\min_{\{T_i\},\{X_j\}} \sum_{i,j} \rho\left(\|x_{ij} - \pi(T_i, X_j)\|^2_{\Sigma}\right)$$

T_i: camera poses, X_j: 3D landmark, $\pi(\cdot)$: projection, ρ: robust loss

구조

- 변수: $n \times 6$-DoF poses + $m \times 3$-DoF landmarks
- Jacobian: $2 \times (6+3)$ per reprojection
- H 구조: “arrow” (poses) + “diagonal” (landmarks)
- Schur complement로 landmark 먼저 eliminate

현대 SfM 파이프라인
COLMAP: SuperPoint feature + Ceres BA
OpenMVG: 고도로 최적화된 sparse BA
NeRF/3DGS: differentiable rendering + BA 결합

부록: 참고문헌 및 추천 자료

교재/리뷰

- Grisetti et al., “A Tutorial on Graph-Based SLAM,” IEEE IS 2010
- Dellaert & Kaess, “Factor Graphs for Robot Perception,” FnTR 2017
- Barfoot, “State Estimation for Robotics,” Cambridge 2017
- Nocedal & Wright, “Numerical Optimization,” Springer 2006

핵심 논문

- Kaess et al., “iSAM2,” IJRR 2012
- Leutenegger et al., “OKVIS,” IJRR 2015
- Qin et al., “VINS-Mono,” TRO 2018
- Triggs et al., “Bundle Adjustment — A Modern Synthesis,” 1999
- Rosinol et al., “Kimera-VIO,” ICRA 2020

부록: 참고문헌 및 추천 자료

교재/리뷰

- Grisetti et al., “A Tutorial on Graph-Based SLAM,” IEEE IS 2010
- Dellaert & Kaess, “Factor Graphs for Robot Perception,” FnTR 2017
- Barfoot, “State Estimation for Robotics,” Cambridge 2017
- Nocedal & Wright, “Numerical Optimization,” Springer 2006

핵심 논문

- Kaess et al., “iSAM2,” IJRR 2012
- Leutenegger et al., “OKVIS,” IJRR 2015
- Qin et al., “VINS-Mono,” TRO 2018
- Triggs et al., “Bundle Adjustment — A Modern Synthesis,” 1999
- Rosinol et al., “Kimera-VIO,” ICRA 2020

오픈소스 코드

- `https://gtsam.org` — GTSAM (Georgia Tech)
- `https://github.com/RainerKuemmerle/g2o` — g2o
- `http://ceres-solver.org` — Ceres (Google)
- `https://github.com/HKUST-Aerial-Robotics/VINS-Mono`
- `https://github.com/MIT-SPARK/Kimera-VIO`

8회차: MSCKF + Visual-Inertial Odometry

지난 시간 한 줄 — 7회차에서 8회차로

필터 = Gauss-Newton의 1회 iter,
IEKF = Gauss-Newton 전체 iter,
MAP = 모든 keyframe을 동시에 최적화.

7회차에서 본 것

- Observability · NIS · Joseph form
- MAP ≡ NLLS · Gauss-Newton/LM · Schur complement
- Bell & Cathey 1993: ESKF ⊂ IEKF ≡ GN on MAP ⊂ Factor Graph
- Marginalization, iSAM2

지난 시간 한 줄 — 7회차에서 8회차로

필터 = Gauss-Newton의 1회 iter,
IEKF = Gauss-Newton 전체 iter,
MAP = 모든 keyframe을 동시에 최적화.

7회차에서 본 것

- Observability · NIS · Joseph form
- MAP ≡ NLLS · Gauss-Newton/LM · Schur complement
- Bell & Cathey 1993: ESKF ⊂ IEKF ≡ GN on MAP ⊂ Factor Graph
- Marginalization, iSAM2

오늘 (8회차) — 한 단계 위로

그 위에 **Sliding Window**를 얹은 MSCKF (Multi-State Constraint KF).
카메라 pose를 "복제(stochastic cloning)"하고, feature uncertainty를
null-space projection으로 algebraic하게 제거 → OpenVINS의 본체.

8회차 로드맵 — MSCKF & Visual-Inertial Odometry

오늘 다룰 내용

1. **파트 1**: MSCKF 구조 및 기초
2. **파트 2**: Null-space projection
3. **파트 3**: Feature triangulation
4. **파트 4**: FEJ (First Estimate Jacobian)
5. **파트 5**: IMU-카메라 캘리브레이션
6. **파트 6**: OpenVINS 전체 파이프라인

핵심 질문

- Feature를 state에 넣지 않고도 VIO가 가능한가?
- Feature depth ambiguity를 어떻게 처리하는가?
- EKF의 consistency 문제를 어떻게 해결하는가?
- IMU와 카메라를 정확히 맞추려면?

Key Reference

Mourikis & Roumeliotis, ICRA 2007
Geneva et al. (OpenVINS), ICRA 2020

MSCKF 등장 배경 — EKF-SLAM의 한계

EKF-SLAM의 문제

- Feature를 state에 포함
- N features $\Rightarrow$ state 차원 $+3N$
- Covariance update: $O(N^2)$
- 실시간 운영 한계 ($N \gtrsim 100$에서 느림)

EKF-SLAM state 구조

$$\boldsymbol{x}_{\text{SLAM}} = \left[\boldsymbol{x}_{\text{IMU}}^\top,\ {}^G\boldsymbol{p}_{f_1}^\top,\ \ldots,\ {}^G\boldsymbol{p}_{f_N}^\top\right]^\top$$

$\Rightarrow 15 + 3N$ 차원, N이 증가하면 폭발적 증가

MSCKF의 해법 (Mourikis & Roumeliotis, ICRA 2007)

- Feature를 state에 **넣지 않음**
- 대신: 여러 카메라 **pose에서 같은 feature**를 보는 기하학적 제약 사용
- State: IMU + 카메라 pose 슬라이딩 윈도우
- Complexity: $O(N_{\text{feat}})$ per update step

핵심 통찰
Feature의 절대 위치가 아닌
다중 뷰 기하학적 제약만 사용

MSCKF 상태 벡터 구조

IMU State (L4 ESKF와 동일)

$$\boldsymbol{x}_{\text{IMU}} = \left[{}^{I}_{G}\boldsymbol{R},\ {}^{G}\boldsymbol{p}_{I},\ {}^{G}\boldsymbol{v}_{I},\ \boldsymbol{b}_{g},\ \boldsymbol{b}_{a}\right] \in \mathrm{SO}(3) \times \mathbb{R}^{12}, \quad \text{error-state: } \mathbb{R}^{15}$$

Camera Clone State (슬라이딩 윈도우)

$$\boldsymbol{x}_{C_i} = \left[{}^{C_i}_{G}\boldsymbol{R},\ {}^{G}\boldsymbol{p}_{C_i}\right] \in \mathrm{SO}(3) \times \mathbb{R}^{3}, \quad \text{error-state: } \mathbb{R}^{6}$$

$i = 1, \ldots, N$: 최근 N개의 카메라 keyframe poses

Full State Vector

$$\boldsymbol{x} = \left[\boldsymbol{x}_{\text{IMU}}^{\top},\ \boldsymbol{x}_{C_1}^{\top},\ \ldots,\ \boldsymbol{x}_{C_N}^{\top}\right]^{\top} \in \mathbb{R}^{15+6N}$$

OpenVINS 기본값

- Sliding window 크기 $N = 10$–20
- $N = 15$: state dim $= 15 + 90 = 105$

주의

Camera pose가 추가될수록 state 증가
$\Rightarrow$ Marginalization으로 관리 필요

Stochastic Cloning — 카메라 Pose 추가

아이디어
IMU propagation 후 새 카메라 frame에 도달할 때,
현재 IMU state에서 카메라 pose를 **복제(clone)**하여 state에 추가

Covariance Augmentation
Clone 전 state $\boldsymbol{x}$, covariance P에서:

$$\boldsymbol{x}^{\text{aug}} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{x}_{C_k} \end{pmatrix}, \qquad P^{\text{aug}} = \begin{pmatrix} P_{xx} & P_{xx}J_c^\top \\ J_c P_{xx} & J_c P_{xx} J_c^\top \end{pmatrix}$$

$J_c \in \mathbb{R}^{6\times 15}$: camera pose $\boldsymbol{x}_{C_k}$의 IMU state $\boldsymbol{x}_{\text{IMU}}$에 대한 Jacobian

왜 "Stochastic" Cloning?

- 단순 복사가 아님
- 기존 state와의 **상관관계 구조**를 정확히 반영
- $P_{xx}J_c^\top$: 새 clone과 기존 state 간 cross-correlation

잘못 구현 시
Cross-correlation 무시 ⇒
Information 손실, inconsistency
(covariance 과소 추정)

Cloning Jacobian J_c 계산

Camera Pose와 IMU State의 관계
Extrinsic calibration: ${}^{C}_{I}T = ({}^{C}_{I}R,\ {}^{I}\boldsymbol{p}_C)$

$$
\begin{aligned}
{}^{C_k}_{G}R &= {}^{C}_{I}R \cdot {}^{I}_{G}R \\
{}^{G}\boldsymbol{p}_{C_k} &= {}^{G}\boldsymbol{p}_I + {}^{I}_{G}R^{\top} \cdot {}^{I}\boldsymbol{p}_C
\end{aligned}
$$

Error-state Jacobian
Rotation 부분 ($\delta\boldsymbol{\phi}_{C_k}$ w.r.t. $\delta\boldsymbol{\phi}_I$):

$$
\delta\boldsymbol{\phi}_{C_k} \approx {}^{C}_{I}R \cdot \delta\boldsymbol{\phi}_I \quad \Rightarrow \quad J_{c,R} = {}^{C}_{I}R \in \mathbb{R}^{3\times 3}
$$

Position 부분 ($\delta\boldsymbol{p}_{C_k}$ w.r.t. $\delta\boldsymbol{\phi}_I,\ \delta\boldsymbol{p}_I$):

$$
\delta\boldsymbol{p}_{C_k} \approx \delta\boldsymbol{p}_I - \left[{}^{I}_{G}R^{\top}\,{}^{I}\boldsymbol{p}_C\right]_{\times}\delta\boldsymbol{\phi}_I \quad \Rightarrow \quad J_{c,p} = \left[-\left[\hat{\boldsymbol{R}}^{\top}\,{}^{I}\boldsymbol{p}_C\right]_{\times},\ \boldsymbol{I}_3\right]
$$

Full $J_c \in \mathbb{R}^{6\times 15}$

$$
J_c = \begin{pmatrix} {}^{C}_{I}R & 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 6} \\ -\left[\hat{\boldsymbol{R}}^{\top}\,{}^{I}\boldsymbol{p}_C\right]_{\times} & \boldsymbol{I}_3 & 0_{3\times 3} & 0_{3\times 6} \end{pmatrix}
$$

(rotation 3, position 3, velocity 3, biases 6 = 15 columns)

Sliding Window 관리 — MSCKF State 진화

MSCKF 한 사이클

1. **IMU Propagation**: $\hat{R}, \hat{p}, \hat{v}, \hat{b}_g, \hat{b}_a$ 및 P 전파
2. **Stochastic Cloning**: 현재 카메라 pose를 state에 복제
3. **Feature Tracking**: KLT optical flow, feature tracks 업데이트
4. **MSCKF Update**: track이 끝난 feature로 update (null-space proj.)
5. **SLAM Update**: 오래 살아남은 feature (선택적)
6. **Marginalization**: 오래된 camera clone 제거

Keyframe vs. All Frames

- 모든 이미지마다 clone할 필요 없음
- 충분한 baseline 기준으로 keyframe 선택
- OpenVINS: 카메라 fps에 맞춰 clone

Marginalization 방법

- **Simple deletion**: oldest clone 그냥 제거 (information 손실, 빠름)
- **Schur complement**: 정확하지만 느림
- OpenVINS: deletion 사용

MSCKF vs. EKF-SLAM 비교

EKF-SLAM

- Feature를 state에 포함
- State 차원: $15 + 3N_{\text{feat}}$
- Covariance update: $O(N_{\text{feat}}^2)$
- Global map: loop closure 가능
- 장기 운영: 일관성 유지 가능
- 단점: large map에서 느림

MSCKF

- Feature를 state에 **넣지 않음**
- State 차원: $15 + 6N_{\text{clone}}$ (고정)
- Update: $O(N_{\text{feat}})$ per step
- No global map $\Rightarrow$ loop closure 어렵
- 단점: drift 누적 (loop closure 없이)
- 장점: 실시간, 메모리 효율

하이브리드: SLAM-augmented MSCKF (OpenVINS 방식)
일부 feature (long-lived) $\Rightarrow$ state에 추가 (EKF-SLAM)
나머지 feature $\Rightarrow$ MSCKF 방식으로 처리
두 가지 방법의 장점을 결합

카메라-IMU Extrinsic: ${}^{C}_{I}T$

정의

$$ {}^{C}_{I}T = \left({}^{C}_{I}R,\ {}^{I}\boldsymbol{p}_{C}\right) \in \mathrm{SE}(3) $$

- ${}^{C}_{I}R$: IMU body frame → Camera frame 회전
- ${}^{I}\boldsymbol{p}_{C}$: IMU frame 내 카메라 원점 위치
- Transformation: ${}^{C}\boldsymbol{p} = {}^{C}_{I}R\,{}^{I}\boldsymbol{p} + {}^{I}\boldsymbol{p}_{C}$ (역방향 주의)

Offline Calibration: Kalibr (ETH)

- AprilGrid target 사용
- IMU + 카메라 동기 데이터 필요
- 정확도: rotation $\sim 0.1°$, translation ~ 0.5 mm

잘못된 Extrinsic의 영향

- Rotation 오차: yaw drift 유발
- Translation 오차: position offset
- 특히 고속 회전 시 크게 악화

Online Calibration

${}^{C}_{I}T$를 EKF state에 포함

⇒ 파트 5에서 상세 설명

측정 모델 — Feature Observation

Pinhole 카메라 모델

Feature j의 3D 위치 ${}^{G}\boldsymbol{p}_f$, 카메라 k의 관측:

$$
{}^{C_k}\boldsymbol{p}_f = {}^{C_k}_{G}R\left({}^{G}\boldsymbol{p}_f - {}^{G}\boldsymbol{p}_{C_k}\right)
$$

$$
\boldsymbol{z}_{jk} = \pi\left({}^{C_k}\boldsymbol{p}_f\right) + \boldsymbol{n}_{jk}, \qquad \pi(\boldsymbol{p}) = \begin{pmatrix} p_x/p_z \\ p_y/p_z \end{pmatrix}
$$

Reprojection Residual

$$
\boldsymbol{r}_{jk} = \boldsymbol{z}_{jk} - \pi\left(\hat{C_k}\boldsymbol{p}_f\right)
$$

Jacobian: $J_\pi = \left.\frac{\partial \pi}{\partial \boldsymbol{p}}\right|_{\hat{p}} = \frac{1}{\hat{p}_z}\begin{pmatrix} 1 & 0 & -\hat{p}_x/\hat{p}_z \\ 0 & 1 & -\hat{p}_y/\hat{p}_z \end{pmatrix} \in \mathbb{R}^{2\times 3}$

6회차에서 배운 것 vs. MSCKF

6회차: 단일 카메라 pose에서의 H matrix

MSCKF: **여러** 카메라 pose에서 동일 feature

$\Rightarrow$ residual stacking + null-space projection

핵심 어려움

${}^{G}\boldsymbol{p}_f$는 알 수 없음 (state에 없음)

$\Rightarrow$ nuisance variable 처리 필요

MSCKF Update 흐름 요약

Feature j: N개 카메라 pose에서 관측
각 pose k에서 2D residual $\boldsymbol{r}_{jk} \in \mathbb{R}^2$:

$$\boldsymbol{r}_{jk} = H_{x_{jk}}\,\delta\boldsymbol{x} + H_{f_{jk}}\,\delta^G\boldsymbol{p}_{f_j} + \boldsymbol{n}_{jk}$$

Stacking all N observations of feature j:

$$\underbrace{\boldsymbol{r}_j}_{2N\times 1} = \underbrace{H_{x_j}}_{2N\times n}\,\delta\boldsymbol{x} + \underbrace{H_{f_j}}_{2N\times 3}\,\delta^G\boldsymbol{p}_{f_j} + \boldsymbol{n}_j$$

MSCKF의 2단계 처리

1. **Triangulation**: ${}^G\hat{\boldsymbol{p}}_{f_j}$ 계산 (feature가 state 없이도 위치 추정)
2. **Null-space projection**: H_{f_j} 제거 → standard EKF update form

핵심 파트 2 미리보기
H_{f_j}의 left null-space $T_j^\perp$ 찾기:
$T_j^{\perp\top} H_{f_j} = 0 \Rightarrow$ feature uncertainty 제거

왜 Null-space Projection이 필요한가?

문제 재확인

$$\boldsymbol{r}_j = H_{x_j}\,\delta\boldsymbol{x} + H_{f_j}\,\delta\boldsymbol{p}_{f_j} + \boldsymbol{n}_j$$

- $\delta\boldsymbol{p}_{f_j}$: feature position uncertainty — **알 수 없음**, state에 없음
- $H_{f_j}\,\delta\boldsymbol{p}_{f_j}$: "nuisance term" — 제거해야 EKF update 가능

해법: Left Null-space Projection
H_{f_j}의 left null-space 행렬 $T_j \in \mathbb{R}^{2N\times(2N-3)}$ 찾기 (s.t. $T_j^\top H_{f_j} = 0$)

$$T_j^\top \boldsymbol{r}_j = T_j^\top H_{x_j}\,\delta\boldsymbol{x} + \underbrace{T_j^\top H_{f_j}}_{=0}\,\delta\boldsymbol{p}_{f_j} + T_j^\top \boldsymbol{n}_j$$

$\Rightarrow$ nuisance term 소거!

결과
$\boldsymbol{r}_j^o = H_j^o\,\delta\boldsymbol{x} + \boldsymbol{n}_j^o$ (standard EKF update form)
여기서 $\boldsymbol{r}_j^o = T_j^\top \boldsymbol{r}_j$, $H_j^o = T_j^\top H_{x_j}$

H_{f_j} 구조 분석

카메라 k에서의 Feature Jacobian

$$H_{f_j}^{(k)} = \frac{\partial \boldsymbol{r}_{jk}}{\partial^G \boldsymbol{p}_{f_j}} = J_\pi \cdot {}^{C_k}_{G}R \in \mathbb{R}^{2\times 3}$$

(J_π: 2×3 projection Jacobian, ${}^{C_k}_{G}R$: rotation matrix 3×3)

Stacked Matrix

$$H_{f_j} = \begin{pmatrix} H_{f_j}^{(1)} \\ H_{f_j}^{(2)} \\ \vdots \\ H_{f_j}^{(N)} \end{pmatrix} \in \mathbb{R}^{2N\times 3}$$

Rank 분석

- Generic case: rank$(H_{f_j}) = 3$
- (3 DOF feature position)
- Left null-space dimension: $2N-3$

Degenerate case

- Pure rotation: rank < 3 가능
- Null-space dimension $< 2N-3$
- $\Rightarrow$ triangulation 실패 신호

Left Null-space 계산 — QR Decomposition

QR Decomposition of H_{f_j}

$$H_{f_j} = Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \qquad \text{(thin QR, } R_1 \in \mathbb{R}^{3\times 3} \text{ upper triangular)}$$

직교 행렬 $Q \in \mathbb{R}^{2N\times 2N}$를 분할:

$$Q = [Q_1 \mid Q_2], \quad Q_1 \in \mathbb{R}^{2N\times 3}, \quad Q_2 \in \mathbb{R}^{2N\times(2N-3)}$$

Left Null-space = Q_2 (열들)

$$T_j = Q_2 \in \mathbb{R}^{2N\times(2N-3)}$$

검증:

$$T_j^\top H_{f_j} = Q_2^\top Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = 0 \quad \checkmark$$

구현

`Eigen::HouseholderQR` or `ColPivHouseholderQR` 사용

Q_2: `qr.householderQ() * Q_block(3:2N)`

Null-space Projection 적용

Step 1: 원래 stacked residual

$$\boldsymbol{r}_j = H_{x_j}\,\delta\boldsymbol{x} + H_{f_j}\,\delta\boldsymbol{p}_{f_j} + \boldsymbol{n}_j, \qquad \boldsymbol{n}_j \sim \mathcal{N}(0,\, \sigma^2 \boldsymbol{I}_{2N})$$

Step 2: $T_j^\top$ 곱하기

$$\underbrace{T_j^\top \boldsymbol{r}_j}_{\boldsymbol{r}_j^o} = \underbrace{T_j^\top H_{x_j}}_{H_j^o}\,\delta\boldsymbol{x} + \underbrace{T_j^\top H_{f_j}}_{=0}\,\delta\boldsymbol{p}_{f_j} + \underbrace{T_j^\top \boldsymbol{n}_j}_{\boldsymbol{n}_j^o}$$

Step 3: 투영된 시스템

$$\boldsymbol{r}_j^o = H_j^o\,\delta\boldsymbol{x} + \boldsymbol{n}_j^o$$

노이즈: $\boldsymbol{n}_j^o \sim \mathcal{N}\left(0,\, \sigma^2 T_j^\top T_j\right) = \mathcal{N}(0,\, \sigma^2 \boldsymbol{I}_{2N-3})$
(직교성: $T_j^\top T_j = \boldsymbol{I}_{2N-3}$ since $T_j = Q_2$)

결과 차원
$\boldsymbol{r}_j^o \in \mathbb{R}^{2N-3}$, $H_j^o \in \mathbb{R}^{(2N-3)\times n}$ (n: total state dim)
Feature 1개당 $2N-3$개의 독립 scalar constraints

Projected H Matrix H_j^o 구조

$H_j^o = T_j^\top H_{x_j}$
$H_{x_j} \in \mathbb{R}^{2N \times n}$: stacked Jacobians w.r.t. all state variables

$$H_{x_j} = \begin{pmatrix} H_{x_j}^{(1)} \\ \vdots \\ H_{x_j}^{(N)} \end{pmatrix}, \quad H_{x_j}^{(k)} = \frac{\partial \boldsymbol{r}_{jk}}{\partial \boldsymbol{x}} = \left[J_\pi \cdot (\text{pose Jacobian w.r.t. clone}_k) \middle| \cdots \right]$$

Sparsity 구조

- Feature j는 N개의 clone pose에만 연결됨
- 나머지 clone에 대한 Jacobian block: 0
- IMU state 블록: IMU state에 대한 Jacobian
- H_j^o is **sparse**: $O(N \cdot n_{\text{clone}})$ nonzeros

Chain rule 요약

$H_{x_j}^{(k)} = J_\pi \cdot \frac{\partial^{C_k} \boldsymbol{p}_f}{\partial \boldsymbol{x}}$
$= J_\pi \cdot \left(\frac{\partial^{C_k} \boldsymbol{p}_f}{\partial \boldsymbol{x}_{C_k}} \cdot \frac{\partial \boldsymbol{x}_{C_k}}{\partial \boldsymbol{x}} + \frac{\partial^{C_k} \boldsymbol{p}_f}{\partial^G \boldsymbol{p}_f} \cdot \frac{\partial^G \boldsymbol{p}_f}{\partial \boldsymbol{x}} \right)$

중요

$\frac{\partial^G \boldsymbol{p}_f}{\partial \boldsymbol{x}} = 0$ (feature는 state에 없음)
$\Rightarrow H_{f_j}$ 항이 별도로 처리됨

EKF Update with H_j^o

모든 Features에 대해 Stack
Feature $j = 1, \ldots, M_f$에 대해:

$$\boldsymbol{r}^o = H^o\,\delta\boldsymbol{x} + \boldsymbol{n}^o, \qquad H^o = \begin{pmatrix} H_1^o \\ \vdots \\ H_{M_f}^o \end{pmatrix} \in \mathbb{R}^{M\times n}$$

여기서 $M = \sum_j (2N_j - 3)$: 전체 projected residual 수

Standard Kalman Update

$$\begin{aligned} S &= H^o P (H^o)^\top + R^o \\ K &= P(H^o)^\top S^{-1} \\ \hat{\boldsymbol{x}} &\leftarrow \hat{\boldsymbol{x}} \oplus K\,\boldsymbol{r}^o \\ P &\leftarrow (\boldsymbol{I} - KH^o)P \end{aligned}$$

$R^o = \sigma^2 \boldsymbol{I}_M$ (identity, from orthogonal projection)

실용 문제
$M \gg n$ (수천 개 features)이면 $S \in \mathbb{R}^{M\times M}$ inversion 비용 $O(M^3)$
$\Rightarrow$ **QR Compression** 필요 (다음 프레임)

QR Compression for Efficiency

아이디어: H^o의 QR 분해

$$H^o = Q \begin{pmatrix} T_H \\ 0 \end{pmatrix}, \quad T_H \in \mathbb{R}^{n \times n} \text{ (upper triangular)}$$

Residual을 같이 변환:

$$Q^\top \boldsymbol{r}^o = \begin{pmatrix} \tilde{\boldsymbol{r}} \\ \text{(discard)} \end{pmatrix}, \quad \tilde{\boldsymbol{r}} \in \mathbb{R}^n$$

압축된 시스템으로 Update

$$\tilde{\boldsymbol{r}} = T_H \, \delta\boldsymbol{x} + \tilde{\boldsymbol{n}}$$

$T_H \in \mathbb{R}^{n \times n}$: $M \times n$ 대신 $n \times n$ 시스템으로 update
Inversion: $O(n^3)$ instead of $O(M \cdot n^2)$

OpenVINS 구현
`UpdaterHelper::measurement_compress_inplace(H_big, res_big)`
내부: `Eigen::HouseholderQR` 사용

OpenVINS UpdaterMSCKF 구조


```
// ov_msckf/src/update/UpdaterMSCKF.cpp
for (auto& feat : feature_vec) {
    MatrixXd H_xj, H_fj;
    VectorXd res_j;
    // Compute H_xj (w.r.t. state) and H_fj (w.r.t. feature)
    UpdaterHelper::get_feature_jacobian_full(
        state, feat, H_xj, H_fj, res_j);
    // QR decomp -> project to null-space of H_fj
    UpdaterHelper::nullspace_project_inplace(H_fj, H_xj, res_j);
    // Stack into big system
    H_big.block(ct, 0, H_xj.rows(), H_big.cols()) = H_xj;
    res_big.segment(ct, res_j.size()) = res_j;
    ct += H_xj.rows();
}
// QR compression: M x n -> n x n
UpdaterHelper::measurement_compress_inplace(H_big, res_big);
// Standard EKF update
StateHelper::EKFUpdate(state, H_big, res_big, R_big);
```

흐름 요약

OpenVINS 내부 — get_feature_jacobian_full


```cpp
// For each camera observation of feature j
for (auto const& [cam_id, uv_obs] : feat.uvs) {
    // Camera pose: R_GtoC, p_CinG
    auto clone = state->_clones_IMU.at(cam_id);
    Matrix3d R_GtoC = clone->Rot();       // ^{C_k}_{G}R
    Vector3d p_CinG = clone->pos();       // ^{G}p_{C_k}

    // Feature in camera frame
    Vector3d p_f_in_C = R_GtoC * (p_f_in_G - p_CinG);

    // 2x3 projection Jacobian
    Matrix<double,2,3> dz_dpc = compute_proj_jac(p_f_in_C);

    // 3x6 Jacobian of p_f_in_C w.r.t. clone pose
    MatrixXd dpc_dxi = compute_pose_jac(R_GtoC, p_f_in_G, p_CinG);

    // 3x3 Jacobian w.r.t. feature position
    Matrix3d dpc_dpf = R_GtoC; // direct rotation

    H_xi += dz_dpc * dpc_dxi;  // 2 x state_dim
    H_fi += dz_dpc * dpc_dpf;  // 2 x 3
}
```

Chain rule 핵심

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = J_\pi \frac{\partial {}^C\boldsymbol{p}_f}{\partial \boldsymbol{x}}, \quad \frac{\partial \boldsymbol{z}}{\partial {}^G\boldsymbol{p}_f} = J_\pi \cdot {}^{C_k}_{G}R$$

Null-space Projection의 기하학적 의미

물리적 해석

- Feature의 **absolute depth**는 단일 뷰에서 rank-deficient
- 여러 뷰에서 depth는 상대적 관계(epipolar constraint)로만 제약됨
- Projected residual: **depth에 독립적인 기하학적 제약**
- 예: 두 카메라에서 본 feature의 bearing 방향 일치 조건

Epipolar 제약과의 관계

2개 카메라 ($N = 2$)일 때:
$2N - 3 = 1$: 단 1개의 scalar constraint
$=$ essential matrix constraint
$\boldsymbol{z}_2^\top E \, \boldsymbol{z}_1 = 0$에 해당

Monocular Scale 문제

Pure monocular VIO: projection이
scale ambiguity를 algebraically eliminate
$\Rightarrow$ scale은 IMU dynamics로만 관측 가능
(gravity + acceleration 활용)

SLAM-augmented MSCKF — 하이브리드 방식

두 종류의 Feature 처리

MSCKF Features

- Track이 끝나면 null-space proj.
- State에 추가되지 않음
- 다수의 feature 처리 가능
- 빠른 update

SLAM Features (Long-lived)

- State에 inverse depth 파라미터로 추가
- 지속적으로 re-observed
- 더 정확한 landmark 추정
- Map consistency 향상

OpenVINS SLAM Feature

- State: $\boldsymbol{x}_f = (\alpha, \beta, \rho)$: inverse depth parameterization
- ${}^{G}\boldsymbol{p}_f = {}^{G}\boldsymbol{p}_{C_{\text{anchor}}} + \frac{1}{\rho}\,{}^{C_{\text{anchor}}}_{G}R^{\top}\,(\alpha, \beta, 1)^{\top}$
- Anchor camera: feature를 처음 본 카메라 clone
- OpenVINS: `state->_features_SLAM`, max 50–100개

파라미터

`max_slam_features`: SLAM feature 최대 수 (default: 50)

MSCKF 한계와 대안

MSCKF 한계

1. **Scale ambiguity**: monocular 카메라만 있을 때 IMU가 있어야 scale 관측 가능
2. **Loop closure 없음**: global map 없음
   ⇒ 장거리에서 drift 누적
3. **Linearization 문제**: 기본 MSCKF는 FEJ 없이 inconsistency 발생 가능
4. **Feature tracking 의존**: poor texture 환경 취약

대안 및 확장

- **SLAM features** (OpenVINS): 일부 feature를 state에
- **Factor graph** (VINS-Mono, ORB-SLAM3 VI): sliding window optimization
- **FEJ** (파트 4): linearization 일관성
- **Stereo**: reliable depth, scale 직접 관측
- **Loop closure**: separate module 연결

실용적 결론

OpenVINS = MSCKF + SLAM features + FEJ + stereo support
⇒ 현재 state-of-the-art open-source VIO (loop closure 제외)

Feature Triangulation 필요성

MSCKF에서 Triangulation이 필요한 이유

1. Feature track이 **끝날 때** 3D position 계산
2. $H_{f_j} = J_\pi \cdot {}^{C_k}_{G}R$: feature 위치 ${}^{G}\hat{\boldsymbol{p}}_f$ 필요
3. Reprojection residual $\boldsymbol{r}_{jk} = \boldsymbol{z}_{jk} - \pi({}^{C_k}_{G}R({}^{G}\hat{\boldsymbol{p}}_f - {}^{G}\boldsymbol{p}_{C_k}))$ 계산에 직접 사용
4. Triangulation 정확도 $\propto$ Jacobian 정확도 $\propto$ update quality

언제 Triangulate?

- Track lost (feature 추적 실패)
- Track length $\geq$ 최소 길이 (보통 3 frames)
- 충분한 parallax 확보 시

주의

Triangulation 실패 $\Rightarrow$
해당 feature는 update에서 제외
(outlier rejection)

Linear Triangulation — DLT

카메라 k에서의 투영 방정식

$$\lambda_k \boldsymbol{z}_k = {}^{C_k}_{G}R\,({}^{G}\boldsymbol{p}_f - {}^{G}\boldsymbol{p}_{C_k}), \quad \boldsymbol{z}_k = (u_k, v_k, 1)^\top \text{ (homogeneous)}$$

DLT: 선형 제약 유도

Cross product ⇒ 각 view에서 2개의 선형 방정식:

$$A_k\,{}^{G}\boldsymbol{p}_f = \boldsymbol{b}_k, \quad A_k \in \mathbb{R}^{2\times 3}$$

Stacking N views: $A\,{}^{G}\boldsymbol{p}_f = \boldsymbol{b}$, $A \in \mathbb{R}^{2N\times 3}$

Least-squares Solution

$${}^{G}\hat{\boldsymbol{p}}_f = (A^\top A)^{-1} A^\top \boldsymbol{b}$$

또는 homogeneous form: last right singular vector of $[A \mid \boldsymbol{b}]$

DLT 한계

수치적으로 불안정, nearly parallel rays 시 ill-conditioned

⇒ Gauss-Newton 초기화용으로만 사용

Inverse Depth 파라미터화

아이디어
Depth d를 직접 추정하지 않고 $\rho = 1/d$ (inverse depth) 사용
$\Rightarrow$ 무한 깊이에서도 수치적으로 안정적

First-camera 기준 파라미터화
Feature의 첫 번째 관측 카메라 C_1에서:

$$\boldsymbol{m}_1 = (u_1, v_1, 1)^\top \text{ (bearing direction in } C_1)$$

$$^G\boldsymbol{p}_f = {}^G\boldsymbol{p}_{C_1} + \frac{1}{\rho}\, {}^G_{C_1}R^\top \boldsymbol{m}_1$$

파라미터: $\boldsymbol{\theta} = (\rho, u_1, v_1) \in \mathbb{R}^3$ (3 DOF)

장점

- Depth uncertainty: uniform in ρ 공간
- Far features: $\rho \to 0$ (수치 안정)
- DLT 초기화 후 Gauss-Newton으로 정밀화

SLAM Feature에도 동일
OpenVINS SLAM feature:
(α, β, ρ) 파라미터로 state에 저장
Anchor = 첫 번째 관측 clone

Nonlinear Triangulation — Gauss-Newton

비용 함수

$$\mathcal{L}(\rho, u_1, v_1) = \sum_{k=1}^{N} \left\| \boldsymbol{z}_k - \pi\big({}^{C_k}\boldsymbol{p}_f(\rho, u_1, v_1)\big) \right\|^2$$

Gauss-Newton 반복

$$J^\top J \, \Delta\boldsymbol{\theta} = -J^\top \boldsymbol{r}, \qquad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$$

$J \in \mathbb{R}^{2N \times 3}$: residual의 $\boldsymbol{\theta}$에 대한 Jacobian

알고리즘

1. DLT로 ${}^{G}\hat{\boldsymbol{p}}_f$ 초기화
2. Inverse depth 변환: $\rho_0 = 1/d_0$
3. GN 반복: 보통 3–5회로 수렴
4. Depth check: $\rho > 0$ 확인

OpenVINS 함수

`FeatureInitializer::`
`single_gaussnewton()`
`single_triangulation_1d()`

Triangulation 유효성 검증

Valid Triangulation 조건 4가지

1. **Minimum parallax**: bearing angle $> \theta_{\text{min}}$ ($\approx 1^\circ$)
2. **Positive depth**: 모든 카메라에서 $p_z > 0$
3. **Reprojection error**: $\|\boldsymbol{z}_k - \pi({}^{C_k}\boldsymbol{p}_f)\| < \epsilon_{\text{max}}$ (1–2 px)
4. **Condition number**: $\kappa(A^\top A) < \kappa_{\text{max}}$

Degenerate 케이스

Pure rotation: baseline 없음
$\Rightarrow$ triangulation 실패, feature 제외

Low baseline/depth: $b/d < 0.02$
$\Rightarrow$ longer track 기다리거나 제외

OpenVINS 파라미터

- `cond_thresh`: condition 임계값
- `max_runs`: GN 최대 반복
- `min_dist`: 최소 depth (m)
- `max_dist`: 최대 depth (m)

OpenVINS Feature Tracking — KLT


```
// TrackKLT: Lucas-Kanade optical flow
// Step 1: Detect new features (Shi-Tomasi)
cv::goodFeaturesToTrack(img_gray, pts_new,
    max_features, 0.01, min_dist);
// Step 2: Track via KLT pyramid
vector<uchar> status;
cv::calcOpticalFlowPyrLK(
    img_prev, img_curr,
    pts_prev, pts_curr, status, err,
    cv::Size(15,15), 3); // win=15x15, 3 levels
// Step 3: RANSAC fundamental matrix outlier removal
cv::findFundamentalMat(
    pts_good_prev, pts_good_curr,
    cv::FM_RANSAC, 1.0, 0.999, inlier_mask);
```

출력 형태

Feature ID → {timestamps, 2D points} track database

Track이 끝나면 triangulation → MSCKF update

Feature Lifetime and Track Management

Track 생명주기

1. **Detection**: Shi-Tomasi / FAST corner 추출
2. **Tracking**: 연속 frame에서 KLT 추적
3. **Loss**: FOV 이탈, occlusion, tracking 실패
4. **Triangulation**: track 종료 시 3D position 계산
5. **Update**: 유효하면 MSCKF update에 사용

Track 파라미터

- 최소 track 길이: 3 frames
- 최대 track 길이: 20–30 frames
- 목표 feature 수: 200–300개
- Grid-based detection: 이미지 균일 분포

OpenVINS 클래스

TrackKLT: KLT tracking
Feature: 개별 feature 정보
FeatureDatabase: 전체 track DB
새 feature: `perform_detection_monocular()`

Triangulation 정확도 분석

정확도에 영향하는 요인

$$\Sigma_{G_{\boldsymbol{p}_f}} \approx \sigma_z^2 \left(J^\top J \right)^{-1}$$

- σ_z: 2D 측정 노이즈 (보통 1 px)
- $J^\top J$: information matrix (baseline, depth, view 수 의존)

핵심 지표

- **Baseline/depth ratio**: $b/d > 0.02$ 이상 권장
- **Condition number**: $\kappa(J^\top J) < 10^3$ 권장
- **View 수**: $N \geq 3$이면 훨씬 안정적

Poor Triangulation 파급 효과

잘못된 ${}^G\hat{\boldsymbol{p}}_f$
$\Rightarrow$ 잘못된 H_{f_j}
$\Rightarrow$ 잘못된 null-space projection
$\Rightarrow$ bad update / inconsistency

완화: Chi-squared Test

$\chi^2 = \boldsymbol{r}^\top R^{-1} \boldsymbol{r} > \chi^2_{0.95}(2N)$
$\Rightarrow$ feature 제외 (outlier)

Outlier Rejection

두 단계 이상치 처리
단계 1 — Frontend (Feature Tracking):

- RANSAC + Fundamental matrix: tracking outlier 제거
- KLT status vector: poor tracking 점 제거

단계 2 — Backend (Update):

- Triangulation 실패 feature 제외
- Chi-squared Mahalanobis distance gate

Mahalanobis Distance Gate

$$\gamma_j = \boldsymbol{r}_j^{o\top} \left(H_j^o P (H_j^o)^\top + R^o \right)^{-1} \boldsymbol{r}_j^o \overset{?}{<} \chi^2_{0.95}(2N_j - 3)$$

통과하지 못하면 feature j의 update 제외

OpenVINS: `chi2_multipler` 파라미터
default = 1.0 (95% threshold), 엄격: 0.5, 관대: 2.0

Stereo MSCKF 확장

Stereo의 이점

- 알려진 baseline b: 직접 depth 계산 가능
- Triangulation 신뢰성 대폭 향상
- Scale 관측: IMU 없이도 metric scale
- Low-texture 환경에서도 강인

Stereo MSCKF H Matrix

Left + Right camera 각각 2D 관측 ($\Rightarrow$ 4D per pose):

$$\boldsymbol{r}_k^{\text{stereo}} = \begin{pmatrix} \boldsymbol{r}_k^L \\ \boldsymbol{r}_k^R \end{pmatrix} \in \mathbb{R}^4, \quad H_k^{\text{stereo}} = \begin{pmatrix} H_k^L \\ H_k^R \end{pmatrix} \in \mathbb{R}^{4\times n}$$

Stacked: $H_f^{\text{stereo}} \in \mathbb{R}^{4N\times 3}$, null-space dim: $4N-3$

실용 하드웨어

- ZED2i (Stereolabs)
- Intel RealSense D435i
- OAK-D (OpenCV AI Kit)

캘리브레이션 필수

Stereo baseline ${}^{C_R}_{C_L}T$
정확도가 성능에 직접 영향
Kalibr로 정밀 캘리브레이션 권장

EKF 선형화 일관성 문제

EKF의 근본적 문제

- EKF는 매 step마다 **현재 추정치**에서 선형화
- 비선형 시스템: linearization point가 step마다 변함
- 서로 다른 linearization point ⇒ **phantom information** 생성
- 증상: covariance가 실제보다 작아짐 (overconfident filter)

VIO에서 특히 심각

- Camera clone: 동일 pose가 여러 번 linearization
- 관측 불가능 방향에서 fictitious information 생성
- ⇒ covariance spuriously shrinks

증상

추정 오차 $> 3\sigma$ bound
NIS > 1 (inconsistent)
⇒ 장기 운영 시 diverge

VIO 관측가능성 — 4개 불관측 방향

이론적 배경 (Hesch et al., TRO 2014)
Monocular VIO 시스템은 **4 DOF 불관측(unobservable)**:

1. **Yaw** (절대 heading): 3D IMU로 절대 방향 측정 불가
2. **Global X, Y, Z position**: 절대 좌표 기준 없음 (GPS 없으면)

Observability matrix $\mathcal{O}$의 null-space dim $= 4$이어야 함

수식

$$\mathcal{O} = \begin{pmatrix} \boldsymbol{H} \\ \boldsymbol{HF} \\ \boldsymbol{HF}^2 \\ \vdots \end{pmatrix} \in \mathbb{R}^{*\times n}, \qquad \mathcal{N}(\mathcal{O}) \overset{?}{=} \mathbb{R}^4$$

표준 EKF의 문제
Linearization point 변화 $\Rightarrow$ $\mathcal{O}$의 null-space 차원 감소
$\Rightarrow$ 관측 불가능한 방향에서 fictitious information $\Rightarrow$ inconsistency

FEJ 정의 — First Estimate Jacobian

FEJ 아이디어 (Li & Mourikis, IJRR 2013)
핵심: Jacobian을 **처음 평가한 추정치에서 고정**

Standard EKF: $\boldsymbol{H}\big|_{\hat{x}_k^+}$ (현재 추정치에서 매번 계산)

FEJ: $\boldsymbol{H}\big|_{\hat{x}^{(1)}}$ (변수의 첫 번째 추정치에서 고정)

FEJ의 보장

- Jacobian이 **일관된** linearization point에서 평가됨
- Observability matrix $\mathcal{O}$의 null-space 차원 보존
- 4개의 불관측 방향으로 fictitious information 생성 방지

FEJ의 단점
첫 번째 추정치가 나쁘면
linearization 오차 누적 가능
⇒ accuracy 약간 감소 가능

Trade-off
Accuracy vs. **Consistency**
실용적으로 FEJ ON이 훨씬
안정적 (consistency 크게 향상)

FEJ 구현 — Linearization Point 관리

구현 절차

1. Camera clone $\boldsymbol{x}_{C_k}$ 가 생성될 때:
   - 현재 추정치 $\hat{\boldsymbol{x}}_{C_k}$: update 시 사용
   - **FEJ 값** $\hat{\boldsymbol{x}}_{C_k}^{(1)}$: Jacobian 계산용으로 별도 저장
2. Prediction step: standard propagation (FEJ와 무관)
3. H matrix 계산: **FEJ 값** 사용
4. State update: 현재 추정치만 업데이트
5. FEJ 값은 **고정**: 이후 업데이트 안 됨

FEJ 적용 범위

- Camera clone poses: FEJ 적용
- IMU state: 매 step 업데이트 (별도 관리)
- SLAM features: anchor pose에 FEJ 적용

OpenVINS 관련 클래스

`PoseJPL::fej()`: FEJ 값 반환
`PoseJPL::set_fej()`: FEJ 설정
`State::_clones_IMU`: clone 관리
Toggle: `_options.do_fej`

OpenVINS에서 FEJ 사용


```
// UpdaterHelper: compute Jacobians with FEJ
for (auto& [cam_id, clone] : state->_clones_IMU) {
    // FEJ value: first estimate when clone was created
    Matrix3d R_fej = clone->Rot_fej();  // fixed rotation
    Vector3d p_fej = clone->pos_fej();  // fixed position
    // Jacobian evaluated at FEJ pose, NOT current estimate
    // -> consistent linearization across all update steps
    MatrixXd H_clone = compute_jacobian(R_fej, p_fej, feat);
}
// Effect: null-space of observability matrix is preserved
// -> 4 unobservable directions remain truly unobservable
// Toggle on/off: state->_options.do_fej (default: true)
```

핵심 포인트

`Rot_fej()` / `pos_fej()`: clone 생성 시점의 추정치 (고정)

⇒ Jacobian이 time-invariant한 point에서 평가됨

FEJ 효과 — 실험 결과 비교

FEJ OFF (표준 EKF)

- Covariance underestimation
- NIS > 1 (inconsistent)
- Yaw 방향 spurious correction 발생
- 장기 운영: heading drift 가속
- EuRoC V3 hard: ATE ~ 1.0 m

FEJ ON

- Consistent covariance
- NIS ≈ 1 (well-calibrated)
- Yaw drift 억제
- 장기 운영: 안정적
- EuRoC V3 hard: ATE ~ 0.3 m

결론

FEJ는 실질적 성능 향상 제공

특히 **고속 회전**과 **장거리 운영**에서 효과 두드러짐

Consistency: 추정 불확실성이 실제 오차를 제대로 반영하는지의 척도

OC-EKF vs. FEJ vs. InEKF 비교

EKF Consistency 개선 방법 3가지

- **OC-EKF** (Huang et al., IJRR 2010):
  H matrix를 사후에 관측가능성 null-space 보존하도록 projection
  ⇒ 수학적으로 엄밀, 구현 복잡
- **FEJ** (Li & Mourikis, IJRR 2013):
  첫 번째 추정치에서 Jacobian 고정
  ⇒ 구현 간단, OC-EKF와 유사한 효과
- **InEKF** (Barrau & Bonnabel, 2017):
  Lie group 위의 Invariant EKF
  ⇒ state-independent Jacobians, 자연적 consistency

실용 현황

- OpenVINS: **FEJ** (default)
- MSCKF-VIO 원본: FEJ 없음
- Contact-InEKF (Michigan): InEKF
- FAST-LIO2: IEKF 반복

이론 vs. 실용

InEKF: 수학적으로 가장 elegant
FEJ: 구현 용이, 매우 실용적
실제 시스템에서 두 방법 모두 효과적

InEKF (Invariant EKF) 간략 소개

핵심 아이디어
Error를 tangent space가 아닌 **Lie group 위에서 직접** 정의:

$$\text{Right Invariant Error:} \quad \eta^r = X^{-1}\hat{X} \in G$$
$$\text{Left Invariant Error:} \quad \eta^l = \hat{X}X^{-1} \in G$$

$X \in G$: true state, $\hat{X}$: estimated state (Lie group 위)

장점: State-independent System Matrix
System matrix $A_t =$ **state에 무관**
⇒ linearization point 변화에도 Jacobian 일정
⇒ FEJ와 동일한 consistency 효과를 자연적으로 달성

관련 시스템

- Contact-aided InEKF (Michigan)
- FAST-LIO2: IEKF 반복 수렴
- LINS: LiDAR-Inertial InEKF

실용적 어려움
IMU+Camera 결합 시:
$SE_{k+1}(3)$ 확장 구조 필요
⇒ 구현 복잡도 증가

공간 캘리브레이션: T_{IC}

정의

$$ {}^{C}_{I}T = ({}^{C}_{I}R,\ {}^{I}\boldsymbol{p}_{C}) \in \mathrm{SE}(3) $$

- ${}^{C}_{I}R$: IMU body frame → Camera frame 회전
- ${}^{I}\boldsymbol{p}_{C}$: IMU frame 내 카메라 원점 위치
- 6 DOF (3 rotation + 3 translation)

Offline Calibration: Kalibr (ETH)

- AprilGrid calibration target
- IMU + Camera 동기 데이터 필요
- Rotation 정확도: $\sim 0.1^{\circ}$
- Translation 정확도: ~ 0.5 mm

잘못된 Extrinsic 영향

- Rotation 오차: yaw drift
- Translation 오차: position offset
- 특히 고속 회전 시 크게 악화

Online Calibration

${}^{C}_{I}T$를 EKF state에 포함

파트 5 계속에서 상세 설명

시간 캘리브레이션: Time Offset t_d

문제: IMU vs. Camera 타임스탬프 불일치

- IMU 타임스탬프: hardware trigger, 매우 정확
- Camera 타임스탬프: software jitter, 노출 시간 기반
- $t_d = t_{\text{cam}} - t_{\text{IMU}}$ (보통 ± 10 ms 범위)

t_d 오차의 영향

잘못된 $t_d \Rightarrow$ IMU state와 카메라 관측 간 **타임 스큐**

$\Rightarrow$ 회전 속도 오차 $\propto \boldsymbol{\omega} \cdot t_d$

정지 상태: 영향 없음 / 고속 회전: 크게 악화

정량적 영향

$\boldsymbol{\omega} = 1$ rad/s, $t_d = 10$ ms:

$\Rightarrow 10$ ms 동안 0.01 rad $\approx 0.57°$ 회전 오차

빠른 드론: $\boldsymbol{\omega} \sim 3$ rad/s $\Rightarrow 1.7°$ 오차

OpenVINS 파라미터

`calib_camimu_dt`: initial guess

`do_calib_camera_timeoffset`: online estimation ON/OFF

Rolling Shutter 효과

Global vs. Rolling Shutter

Global Shutter

- 모든 row 동시 노출
- 산업용 카메라, event 카메라
- 고속 motion에서 선명

Rolling Shutter

- Row마다 노출 시간 차이 (t_r: readout time)
- 일반 CMOS 카메라 (스마트폰 포함)
- 고속 motion에서 이미지 왜곡

Rolling Shutter 보정

Row i의 실제 노출 시간: $t_i = t_{\text{frame}} + i \cdot t_r / N_{\text{rows}}$
보정: 각 row의 노출 시간에서 IMU state 보간

$$\hat{T}_{\text{cam}}(t_i) = \text{interpolate}(\hat{T}_{\text{cam}}(t_{\text{frame}}),\ \hat{T}_{\text{cam}}(t_{\text{frame}} + t_r))$$

OpenVINS 지원
`CameraData::is_fisheye`, rolling shutter 모델 포함
`_cam_RDRH`: readout time 파라미터

Intrinsic Camera Calibration

Pinhole 모델 파라미터

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

f_x, f_y: focal length (pixels), c_x, c_y: principal point

Distortion 모델

Radial-Tangential
k_1, k_2, p_1, p_2
일반 카메라 (RealSense 등)

Equidistant (fisheye)
k_1, k_2, k_3, k_4
광각 카메라 (180° FOV)

두 가지 처리 방법

- **Undistortion**: 이미지 전처리 후 pinhole 사용
- **On-the-fly**: Jacobian에 distortion 항 포함
- OpenVINS: 두 방법 모두 지원

OpenVINS 카메라 모델

`PinholeRadtan`: 방사/접선 왜곡
`PinholeEqui`: fisheye (equidist)

Kalibr 캘리브레이션 도구

Kalibr (ETH Zurich) 사용법

```
# IMU-Camera extrinsic + time offset calibration
kalibr_calibrate_imu_camera \
  --target aprilgrid.yaml \
  --imu imu.yaml \
  --cams cam.yaml \
  --bag calibration.bag \
  --show-extraction

# Multi-camera calibration
kalibr_calibrate_cameras \
  --target aprilgrid.yaml \
  --bag calibration.bag \
  --models pinhole-radtan pinhole-radtan \
  --topics /cam0/image_raw /cam1/image_raw
```

출력 결과

- ${}^{C}_{I}T$: IMU-camera extrinsic (회전 + 이동)
- t_d: camera-IMU time offset
- Intrinsics + distortion coefficients

일반적 정확도
Rotation: $\sim 0.1°$, Translation: ~ 0.5 mm, t_d: ~ 0.5 ms

Online T_{IC} Calibration in EKF

State 확장

$$\boldsymbol{x}_{\text{ext}} = [{}^{C}_{I}\boldsymbol{\phi},\ {}^{I}\boldsymbol{p}_{C}] \in \mathbb{R}^{6} \quad \text{(6DOF extrinsic error-state)}$$

Full state: $\boldsymbol{x} = [\boldsymbol{x}_{\text{IMU}}^{\top},\ \boldsymbol{x}_{C_1}^{\top},\ \ldots,\ \boldsymbol{x}_{\text{ext}}^{\top}]^{\top}$

Dynamics
Extrinsic은 정적 (시간 불변): $\dot{\boldsymbol{x}}_{\text{ext}} = 0$
Prediction step: identity (covariance 전파만)

Update
모든 camera factor의 H matrix에 $\boldsymbol{x}_{\text{ext}}$에 대한 항 추가:

$$H_{\text{ext}} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}_{\text{ext}}} = J_{\pi} \cdot \frac{\partial {}^{C_k}\boldsymbol{p}_f}{\partial {}^{C}_{I}T}$$

OpenVINS 구현
`state->_calib_IMUtoCAM`: extrinsic calibration state
`do_calib_camera_pose`: online calibration ON/OFF

Online t_d Calibration

State 확장
$t_d \in \mathbb{R}$ (scalar): camera-IMU time offset

Residual의 t_d 의존성
카메라 관측 시각 t_{img}, IMU state $t_{\text{img}} - t_d$에서 평가:

$$\boldsymbol{r}(t_d) = \boldsymbol{z} - h\big(\hat{\boldsymbol{x}}(t_{\text{img}} - t_d)\big)$$

Jacobian:

$$\frac{\partial \boldsymbol{r}}{\partial t_d} = \frac{\partial h}{\partial \boldsymbol{x}} \cdot \frac{d\hat{\boldsymbol{x}}}{dt}\bigg|_{t_{\text{img}} - t_d}$$

즉: IMU 각속도/속도 $\boldsymbol{\omega}, \boldsymbol{v}$가 t_d Jacobian에 곱해짐

관측 불가 조건
정지 상태: $\boldsymbol{v} \approx 0, \boldsymbol{\omega} \approx 0$
$\Rightarrow$ t_d Jacobian ≈ 0
$\Rightarrow$ 정지 상태에서 t_d 추정 불가

OpenVINS 파라미터
`do_calib_camera_timeoffset`: online t_d estimation ON/OFF
초기값: `calib_camimu_dt`

Multi-Camera 확장

OpenVINS 다중 카메라 지원

- 최대 5개 카메라까지 지원
- 각 카메라별 별도 ${}^{C}_{I}T$ extrinsic
- 각 카메라별 별도 intrinsics + distortion
- Stereo: fixed baseline $\Rightarrow$ reliable depth

Multi-camera VIO 이점

- 관측가능성 크게 향상 (더 많은 constraint)
- Degenerate motion (pure rotation) 대처 능력 향상
- Wide field of view: feature tracking 지속성 증가
- Stereo: monocular scale ambiguity 해결

실용 예시

하드웨어	카메라	IMU
ZED2i	Stereo RGB	내장 6-DOF
RealSense D435i	Stereo IR + RGB	BMI055
OAK-D	Stereo + RGB	내장
KAIST sensor	Stereo	Xsens MTI-300

IMU Pre-processing 및 초기화

IMU 사전 처리

- Rate: 200–1000 Hz (고율일수록 integration 정확)
- Synchronization: camera trigger로 동기화 or PPS pulse
- Allan deviation: IMU noise 특성 분석에 사용
  - σ_g: gyroscope noise (rad/s/$\sqrt{\text{Hz}}$)
  - σ_a: accelerometer noise (m/s^2/$\sqrt{\text{Hz}}$)

OpenVINS Static Initializer

정지 상태에서 초기화:

1. 가속도 분산으로 정지 감지: var($\boldsymbol{a}_m$) $< \sigma^2_{\text{thresh}}$
2. Gravity 방향 추정: $\hat{\boldsymbol{g}} = -\bar{\boldsymbol{a}}_m / \|\bar{\boldsymbol{a}}_m\|$
3. 초기 roll, pitch 추정 (yaw = 0 assumed)
4. Gyro/accel bias 초기 추정

Dynamic Initializer

최소 0.5초 dynamic motion $\Rightarrow$ velocity + gravity + bias 동시 추정

캘리브레이션 파라미터 영향 분석

파라미터별 영향

파라미터	오차	영향
${}^{C}_{I}R$ (rotation)	$0.5°$	yaw drift 유발
${}^{I}\boldsymbol{p}_{C}$ (translation)	1 cm	position offset
t_d (time offset)	10 ms	rotation error $\propto \boldsymbol{\omega} \cdot t_d$
f_x, f_y (focal length)	1 px	scale error
k_1, k_2 (distortion)	0.01	edge feature 왜곡

Best Practice

1. Kalibr로 offline 캘리브레이션 먼저 수행
2. OpenVINS로 online refinement (extrinsic, t_d)
3. Calibration 결과 검증: APE/RPE on EuRoC
4. 정기적으로 재캘리브레이션 (온도 변화 등)

주의

Online calibration: 충분한 excitation 필요
정지/직선 주행만으로는 수렴 보장 어려움

OpenVINS 아키텍처 개요

4개 주요 모듈

1. **ov_core**: 기본 타입, feature tracking (KLT, ORB), 수학 유틸
2. **ov_msckf**: MSCKF estimator core (state, update, propagation)
3. **ov_init**: visual-inertial initializer (static + dynamic)
4. **ov_eval**: evaluation tools (APE, RPE, consistency analysis)

ROS 지원

- ROS1 Noetic
- ROS2 Humble / Foxy
- Launch file: `subscribe.launch`
- 주요 topic: `/ov_msckf/odomimu`

GitHub
github.com/rpng/open_vins
MIT License (완전 오픈소스)
Geneva et al., ICRA 2020

Frontend: Feature Tracking


```
// TrackKLT::feed_new_camera()
// 1. Detect new features (Shi-Tomasi)
cv::goodFeaturesToTrack(img, pts_new, max_features,
                        0.01, min_dist);
// 2. Track existing features (KLT pyramid)
cv::calcOpticalFlowPyrLK(img_prev, img_curr,
    pts_prev, pts_curr, status, err, cv::Size(15,15), 3);
// 3. Outlier rejection (RANSAC fundamental matrix)
cv::findFundamentalMat(
    pts0, pts1, cv::FM_RANSAC, 1.0, 0.999, mask);
// Stereo: left-right matching for immediate depth
if (stereo) {
    match_stereo(img_left, img_right,
                 pts_left, pts_right);
}
```

출력
Feature ID → {timestamps, 2D positions} track database
Track이 끝나면 MSCKF update 또는 SLAM feature로 등록

Backend: MSCKF Estimator 흐름

VioManager 한 사이클 (카메라 frame 도착 시)

1. `feed_measurement_imu()`: IMU propagation
   - RK4 integration of IMU kinematics
   - $\hat{R}, \hat{p}, \hat{v}, \hat{b}_g, \hat{b}_a$ 및 covariance P 전파
2. `feed_measurement_camera()`:
   - 2.1 Feature track 업데이트 (frontend)
   - 2.2 Stochastic cloning (카메라 pose 추가)
   - 2.3 **MSCKF update** (lost/long features → null-space proj)
   - 2.4 **SLAM update** (long-lived features in state)
   - 2.5 State marginalization (oldest clone 제거)

핵심 클래스

`VioManager`: 최상위 관리
`UpdaterMSCKF`: MSCKF update
`UpdaterSLAM`: SLAM feature update
`Propagator`: IMU 전파

Initialization (정적 + 동적)

Static Initializer

- 정지 상태 감지:
  $\text{var}(\boldsymbol{a}_m) < \sigma^2_{\text{thr}}$
- Gravity 방향 추정:
  $\hat{\boldsymbol{g}}_0 = -\bar{\boldsymbol{a}}_m / \|\bar{\boldsymbol{a}}_m\|$
- 초기 roll, pitch 계산
  (yaw $= 0$ assumed)
- Bias 초기값 추정:
  $\hat{\boldsymbol{b}}_g = \bar{\boldsymbol{\omega}}_m$

초기화 실패 케이스
정지 상태에서 dynamic init 시도 $\Rightarrow$ 실패
Low-texture 환경 $\Rightarrow$ visual SfM 실패
해결: static init fallback 사용

Dynamic Initializer

- 최소 0.5초 dynamic motion 필요
- Camera + IMU alignment
- Velocity + gravity + bias 동시 추정
- Visual SfM으로 scale 초기화
- 더 정확하지만 시간 필요

성능 평가 — EuRoC Dataset

EuRoC MAV Benchmark

- 11개 시퀀스: V1/V2/V3 (easy/medium/hard), MH01–05
- Metric: ATE (Absolute Trajectory Error, RMSE)
- Ground truth: Vicon mocap system

OpenVINS Stereo 성능 (ATE, m)

Sequence	OpenVINS	VINS-Mono	ORB-SLAM3 VI
V1_01 (easy)	0.05	0.09	0.04
V2_02 (medium)	0.08	0.12	0.06
V1_03 (hard)	0.25	0.38	0.08
MH_04 (medium)	0.10	0.15	0.09

결론

OpenVINS: 오픈소스 + 실시간 + ROS 지원 + 경쟁력 있는 정확도
ORB-SLAM3: loop closure 있어 hard sequence에서 더 좋음

OpenVINS vs. VINS-Mono vs. ORB-SLAM3 VI

VIO 시스템 비교

시스템	방식	Loop	Stereo	라이선스
OpenVINS	MSCKF+SLAM	×	✓	GPLv3
VINS-Mono	Sliding win opt	✓	✓	GPL
ORB-SLAM3 VI	Feature-based opt	✓	✓	GPL
KIMERA-VIO	GTSAM-based	✓(mesh)	✓	BSD
OKVIS2	Nonlinear opt	×	✓	BSD

OpenVINS 강점

- 가장 활발한 community
- 상세한 문서 + tutorial
- FEJ, online calib 지원
- Multi-camera 지원

단점

Loop closure 없음
⇒ 대규모 환경에서 drift
⇒ SLAM 시스템 필요 시:
VINS-Fusion 또는 ORB-SLAM3

실전 파라미터 튜닝 가이드

주요 파라미터

파라미터	범위	설명
`max_cameras`	1–5	카메라 수
`max_slam_features`	50–100	SLAM feature 수
`num_pts`	200–400	tracked feature 수
`clone_states_num`	10–20	sliding window 크기
`calib_camimu_dt`	± 0.05 s	time offset 초기값
`chi2_multipler`	0.5–2.0	outlier rejection threshold
`up_msckf_sigma_px`	1–2	feature noise (pixels)

IMU Noise 파라미터 측정 방법

Allan deviation analysis: 정지 상태에서 IMU 1시간 기록
σ_g (gyro noise), σ_a (accel noise),
σ_{bg} (gyro random walk), σ_{ba} (accel random walk)

튜닝 순서

1. IMU noise params (Allan) $\rightarrow$ 2. Camera intrinsics (Kalibr)
$\rightarrow$ 3. Extrinsic T_{IC} (Kalibr) $\rightarrow$ 4. Run + evaluate ATE/RPE

요약: MSCKF + VIO 핵심 5가지

오늘 배운 핵심 개념

1. **MSCKF**: Feature 없이 카메라 pose sliding window
   $O(N)$ complexity, EKF-SLAM 대비 효율적
2. **Stochastic Cloning**: correlation-preserving augmentation
   P^{aug}에 cross-correlation 정확히 반영
3. **Null-space Projection**: feature depth ambiguity algebraic elimination
   $T_j^\top H_{f_j} = 0 \Rightarrow$ standard EKF update form
4. **FEJ**: observability-consistent linearization
   첫 번째 추정치에서 Jacobian 고정 $\Rightarrow$ 4 unobservable DOF 보존
5. **Online Calibration**: T_{IC}, t_d, intrinsics in EKF state
   충분한 excitation 하에서 자동 refinement

다음 회차 예고

9회차: FAST-LIO2 (LiDAR-Inertial IEKF)
MSCKF $\rightarrow$ point cloud 기반으로 확장
ikd-Tree, incremental mapping, 고속 LiDAR odometry

VIO 실패 원인 분석

일반적인 실패 원인 4가지

1. **과도한 drift**: poor feature tracking, degenerate motion
   $\Rightarrow$ feature 수 늘리기, 최소 parallax 확보
2. **Filter inconsistency**: 잘못된 noise 파라미터, bad initialization
   $\Rightarrow$ IMU noise 재측정 (Allan deviation), FEJ 활성화
3. **Numerical crash**: singular H matrix, ill-conditioned triangulation
   $\Rightarrow$ chi2 threshold 강화, triangulation depth check 강화
4. **Scale drift**: 오직 monocular + 약한 IMU excitation
   $\Rightarrow$ stereo 사용 또는 loop closure 추가

Diagnostic Checklist

- Feature track lifetime histogram (너무 짧으면 texture 부족)
- Covariance growth rate (너무 빠르면 noise params 문제)
- NIS (Normalized Innovation Squared): ≈ 1이 정상
- ATE on known sequence (EuRoC) 먼저 검증

운동 퇴화 (Degenerate Motion)

퇴화 운동 종류

1. **Pure Rotation**: baseline 없음 ⇒ triangulation 실패
   - IMU rotation에 의존, translation 추정 불가
   - 해결: gyroscope만으로 short-term rotation 유지
2. **Pure Translation along camera axis**: epipolar 퇴화
   - Feature bearing 변화 없음 ⇒ no optical flow
3. **Low-texture environment**: feature 부족
   - 해결: IMU 의존도 높이기, LiDAR 추가

Scale Unobservability

Pure rotation sequence:
scale drift 급격히 증가
⇒ IMU acceleration 없으면
scale 완전 unobservable

실용 해결책

- 의도적으로 translation 포함한 운동
- Stereo 카메라 사용
- Wheel odometry 추가

진단 도구 — RViz 시각화 및 평가

RViz 시각화

- Trajectory: `/ov_msckf/odomimu` (nav_msgs/Odometry)
- MSCKF features: `/ov_msckf/features_msckf` (sensor_msgs/PointCloud2)
- SLAM features: `/ov_msckf/features_slam`
- Covariance ellipsoid (position uncertainty)

정량적 평가: ov_eval 패키지

- **ATE** (Absolute Trajectory Error): 전역 정확도
- **RPE** (Relative Pose Error): 국소 정확도, drift rate
- **NEES** (Normalized Estimation Error Squared): consistency
- **NIS** (Normalized Innovation Squared): update quality

평가 명령 예시

`rosrun ov_eval plot_trajectories`
`rosrun ov_eval error_comparison` (vs. ground truth)

참고 자료 및 코드

핵심 논문

- **MSCKF**: A. Mourikis & S. Roumeliotis, “A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation,” ICRA 2007
- **FEJ**: M. Li & A. Mourikis, “High-precision, consistent EKF-based visual-inertial odometry,” IJRR 2013
- **Observability**: J.A. Hesch et al., “Camera-IMU-based localization: Observability analysis and consistency improvement,” TRO 2014
- **OpenVINS**: P. Geneva et al., “OpenVINS: A Research Platform for Visual-Inertial Estimation,” ICRA 2020

오픈소스 코드

- **OpenVINS**: `github.com/rpng/open_vins` (GPLv3)
- **Kalibr**: `github.com/ethz-asl/kalibr` (BSD)
- **VINS-Mono**: `github.com/HKUST-Aerial-Robotics/VINS-Mono` (GPL)
- **ORB-SLAM3**: `github.com/UZ-SLAMLab/ORB_SLAM3` (GPL)

9회차: LiDAR 상태 추정 — FAST-LIO / FAST-LIVO

지난 시간 한 줄 — 8회차에서 9회차로

카메라가 IEKF ≡ GN의 한 정점이라면,
LiDAR는 또 하나의 정점.

8회차에서 본 것 (Visual-Inertial)

- Stochastic cloning · Null-space projection · FEJ
- 약한 비선형 (Reprojection) → MSCKF 1회 update로 충분
- OpenVINS 코드 전체 매핑

지난 시간 한 줄 — 8회차에서 9회차로

카메라가 IEKF ≡ GN의 한 정점이라면,
LiDAR는 또 하나의 정점.

8회차에서 본 것 (Visual-Inertial)

- Stochastic cloning · Null-space projection · FEJ
- 약한 비선형 (Reprojection) → MSCKF 1회 update로 충분
- OpenVINS 코드 전체 매핑

오늘 (9회차) — LiDAR-Inertial
LiDAR Point-to-plane은 강한 비선형이라 EKF 1회로는 부족 →
IEKF on SO(3) (FAST-LIO/2), 즉 **Gauss-Newton 전체 iter**가 필수.
그 위에 **ikd-tree** 동적 맵 + **per-point undistortion** = FAST-LIO의 정수.

9회차 로드맵

오늘 강의 구성

1. **파트 1**: LiDAR 기초 + 포인트 클라우드 처리
2. **파트 2**: LiDAR Odometry 기초 — ICP, LOAM
3. **파트 3**: LIO-SAM — Factor Graph LiDAR-IMU
4. **파트 4**: FAST-LIO / FAST-LIO2 — IEKF on Manifold
5. **파트 5**: FAST-LIVO / FAST-LIVO2 — LiDAR+IMU+Camera
6. **파트 6**: 현장 적용 및 시스템 비교

9회차 로드맵

오늘 강의 구성

1. **파트 1**: LiDAR 기초 + 포인트 클라우드 처리
2. **파트 2**: LiDAR Odometry 기초 — ICP, LOAM
3. **파트 3**: LIO-SAM — Factor Graph LiDAR-IMU
4. **파트 4**: FAST-LIO / FAST-LIO2 — IEKF on Manifold
5. **파트 5**: FAST-LIVO / FAST-LIVO2 — LiDAR+IMU+Camera
6. **파트 6**: 현장 적용 및 시스템 비교

핵심 질문
LiDAR 측정을 ESKF에 어떻게 통합하는가?
Point-to-plane residual의 Jacobian은 어떻게 유도되는가?

LiDAR 센서 종류

Spinning LiDAR

- Velodyne HDL-64E (64채널)
- Ouster OS1-64/128
- 360° 수평 FOV
- 10Hz scan rate
- Range: 최대 120m

Solid-state LiDAR

- Livox Avia ($\approx 70°$ FOV, rosette)
- Livox MID-360 ($\approx 360°$ 반구형)
- 비반복 스캔 패턴
- 기계적 고장 없음

주요 특성

- Range: 10–200 m
- Accuracy: 2–5 cm
- Per-point timestamp (spinning)
- Intensity 정보 포함

FAST-LIO 지원
Spinning + Solid-state 모두 지원
Livox SDK 통합

포인트 클라우드 표현

Raw Scan 구조

- N개의 3D 포인트 $\{p_i\} \subset \mathbb{R}^3$
- 각 포인트: timestamp t_i (spinning LiDAR)
- Attributes: range r_i, intensity s_i, return count

포인트 클라우드 표현

Raw Scan 구조

- N개의 3D 포인트 $\{p_i\} \subset \mathbb{R}^3$
- 각 포인트: timestamp t_i (spinning LiDAR)
- Attributes: range r_i, intensity s_i, return count

Organized Cloud

- 2D 배열: (ring, angle)
- 구조적 근접성 활용
- Range image 형태

Unorganized Cloud

- Raw point list
- kd-tree 탐색 필요
- FAST-LIO2 방식

포인트 클라우드 표현

Raw Scan 구조

- N개의 3D 포인트 $\{p_i\} \subset \mathbb{R}^3$
- 각 포인트: timestamp t_i (spinning LiDAR)
- Attributes: range r_i, intensity s_i, return count

Organized Cloud

- 2D 배열: (ring, angle)
- 구조적 근접성 활용
- Range image 형태

Unorganized Cloud

- Raw point list
- kd-tree 탐색 필요
- FAST-LIO2 방식

좌표계 관례
LiDAR 좌표계: right-hand,
X forward, Y left, Z up (또는 제조사별 상이)

Undistortion (운동 보상)

문제: Motion Distortion

Spinning LiDAR: 한 scan (≈ 100 ms) 동안 센서 이동

$\Rightarrow$ 각 포인트가 서로 다른 위치에서 측정됨 $\rightarrow$ 왜곡

Undistortion (운동 보상)

문제: Motion Distortion
Spinning LiDAR: 한 scan (≈ 100 ms) 동안 센서 이동
$\Rightarrow$ 각 포인트가 서로 다른 위치에서 측정됨 $\rightarrow$ 왜곡

해결: Per-point 자세 보간
각 포인트 p_k를 scan 종료 시점 frame으로 보정:

$$p_k^{\text{end}} = R_{\text{end}}^k \, p_k^k + t_{\text{end}}^k$$

여기서 R_{end}^k, t_{end}^k는 IMU 적분으로 얻은 상대 자세

Undistortion (운동 보상)

문제: Motion Distortion
Spinning LiDAR: 한 scan (≈ 100 ms) 동안 센서 이동
$\Rightarrow$ 각 포인트가 서로 다른 위치에서 측정됨 $\rightarrow$ 왜곡

해결: Per-point 자세 보간
각 포인트 p_k를 scan 종료 시점 frame으로 보정:

$$p_k^{\text{end}} = R_{\text{end}}^k \, p_k^k + t_{\text{end}}^k$$

여기서 R_{end}^k, t_{end}^k는 IMU 적분으로 얻은 상대 자세

IMU 기반 보간

- IMU 200Hz 적분
- 각 포인트 timestamp에 보간
- SLERP (rotation) + 선형 (translation)

FAST-LIO 방식
Undistortion을 IEKF 내부에 통합
별도 전처리 단계 불필요

Voxel Grid Downsampling

목적

- 입력: dense point cloud (수만 개)
- 각 voxel에서 centroid 1개 선택
- 계산 효율: $O(N) \rightarrow O(N/k)$

Voxel Grid Downsampling

목적

- 입력: dense point cloud (수만 개)
- 각 voxel에서 centroid 1개 선택
- 계산 효율: $O(N) \rightarrow O(N/k)$

PCL 예시

```
pcl::VoxelGrid<pcl::PointXYZI> vg;
vg.setInputCloud(cloud_in);
vg.setLeafSize(0.2f, 0.2f, 0.2f); // 20cm voxel
vg.filter(*cloud_filtered);
```

Voxel Grid Downsampling

목적

- 입력: dense point cloud (수만 개)
- 각 voxel에서 centroid 1개 선택
- 계산 효율: $O(N) \rightarrow O(N/k)$

PCL 예시

```
pcl::VoxelGrid<pcl::PointXYZI> vg;
vg.setInputCloud(cloud_in);
vg.setLeafSize(0.2f, 0.2f, 0.2f); // 20cm voxel
vg.filter(*cloud_filtered);
```

Voxel 크기 선택

- 실내: 0.1–0.2 m
- 실외: 0.3–0.5 m
- 환경 밀도에 따라 조정

FAST-LIO2
ikd-tree에 voxelization 통합
별도 필터링 불필요

ICP — Iterative Closest Point 기초

문제 정의
Source cloud A를 target cloud B에 align하는 변환 $T \in \mathrm{SE}(3)$ 탐색

$$T^* = \arg\min_T \sum_i \| p_i^B - T\, p_i^A \|^2$$

ICP — Iterative Closest Point 기초

문제 정의
Source cloud A를 target cloud B에 align하는 변환 $T \in \mathrm{SE}(3)$ 탐색

$$T^* = \arg\min_T \sum_i \| p_i^B - T\, p_i^A \|^2$$

ICP 알고리즘

1. Nearest neighbor search (kd-tree): $p_i^A \leftrightarrow p_j^B$
2. Correspondences로 T 추정 (SVD)
3. T 적용 후 반복 → 수렴까지

ICP — Iterative Closest Point 기초

문제 정의
Source cloud A를 target cloud B에 align하는 변환 $T \in \mathrm{SE}(3)$ 탐색

$$T^* = \arg\min_T \sum_i \| p_i^B - T\, p_i^A \|^2$$

ICP 알고리즘

1. Nearest neighbor search (kd-tree): $p_i^A \leftrightarrow p_j^B$
2. Correspondences로 T 추정 (SVD)
3. T 적용 후 반복 → 수렴까지

한계

- Local minimum 존재
- Initial guess 필요
- Outlier에 민감

Point-to-Plane ICP
Surface normal 활용
더 빠른 수렴
FAST-LIO 방식과 유사

GICP — Generalized ICP

아이디어: 포인트에 covariance 부여
각 포인트 p_i에 covariance Σ_i (표면 방향 반영):

$$T^* = \arg\min_T \sum_i (p_i^B - T p_i^A)^\top \left(\Sigma_i^B + T\Sigma_i^A T^\top\right)^{-1} (p_i^B - T p_i^A)$$

GICP — Generalized ICP

아이디어: 포인트에 covariance 부여
각 포인트 p_i에 covariance Σ_i (표면 방향 반영):

$$T^* = \arg\min_T \sum_i (p_i^B - Tp_i^A)^\top \left(\Sigma_i^B + T\Sigma_i^A T^\top\right)^{-1} (p_i^B - Tp_i^A)$$

NDT (Normal Distributions Transform)

- 각 voxel에 Gaussian fitting
- $p(x) = \mathcal{N}(\mu_v, \Sigma_v)$
- 자동차 자율주행에서 인기

FAST-LIO2 방식

- Point-to-plane
- Nearest neighbor + normal
- Covariance: $R = \sigma^2 I$
- 단순하지만 효과적

Feature Extraction for LiDAR

LOAM-style Features
포인트 곡률(curvature) 기반 분류:

$$c = \frac{1}{|S| \, \|\boldsymbol{p}_i\|} \left\| \sum_{j \in S} (\boldsymbol{p}_j - \boldsymbol{p}_i) \right\|$$

- **Edge features**: 높은 곡률 (모서리, 가장자리)
- **Planar features**: 낮은 곡률 (바닥, 벽)

Feature Extraction for LiDAR

LOAM-style Features
포인트 곡률(curvature) 기반 분류:

$$c = \frac{1}{|S| \, \|\boldsymbol{p}_i\|} \left\| \sum_{j \in S} (\boldsymbol{p}_j - \boldsymbol{p}_i) \right\|$$

- **Edge features**: 높은 곡률 (모서리, 가장자리)
- **Planar features**: 낮은 곡률 (바닥, 벽)

LOAM 방식의 한계

- Featureless 환경에서 실패
- Parameter 튜닝 필요
- 불규칙 패턴에 민감

트렌드: Direct Methods
FAST-LIO2: feature extraction 없음
Raw point → plane fitting
더 범용적, 더 빠름

Map 표현 방식

주요 Map 자료구조

1. **Point cloud map**: 단순, 메모리 많음, 검색 $O(N)$
2. **Voxel map**: 정해진 해상도, occupancy grid
3. **ikd-tree** (FAST-LIO2): incremental kd-tree, dynamic insert/delete
4. **iVox** (Faster-LIO): hash-based voxel, $O(1)$ insert

Map 표현 방식

주요 Map 자료구조

1. **Point cloud map**: 단순, 메모리 많음, 검색 $O(N)$
2. **Voxel map**: 정해진 해상도, occupancy grid
3. **ikd-tree** (FAST-LIO2): incremental kd-tree, dynamic insert/delete
4. **iVox** (Faster-LIO): hash-based voxel, $O(1)$ insert

핵심 관찰
Map 효율이 LiDAR odometry 속도의 핵심
kd-tree rebuild $\rightarrow$ ikd-tree incremental update = 2–3배 속도 향상

Map 표현 방식

주요 Map 자료구조

1. **Point cloud map**: 단순, 메모리 많음, 검색 $O(N)$
2. **Voxel map**: 정해진 해상도, occupancy grid
3. **ikd-tree** (FAST-LIO2): incremental kd-tree, dynamic insert/delete
4. **iVox** (Faster-LIO): hash-based voxel, $O(1)$ insert

핵심 관찰
Map 효율이 LiDAR odometry 속도의 핵심
kd-tree rebuild $\rightarrow$ ikd-tree incremental update = 2–3배 속도 향상

선택 기준

- 실시간 요구: ikd-tree 또는 iVox
- 정밀 mapping: dense point cloud map
- 메모리 제한: voxel map

LiDAR 측정 노이즈 모델

노이즈 특성

- Range noise: $\sigma_r \approx 0.02\text{–}0.05$ m (Gaussian)
- Angle noise: $\sigma_\theta \approx 0.01°$
- 3D 포인트 불확도: $\sigma_p \approx r\,\sigma_\theta$ (먼 포인트일수록 불정확)

LiDAR 측정 노이즈 모델

노이즈 특성

- Range noise: $\sigma_r \approx 0.02\text{–}0.05$ m (Gaussian)
- Angle noise: $\sigma_\theta \approx 0.01°$
- 3D 포인트 불확도: $\sigma_p \approx r\,\sigma_\theta$ (먼 포인트일수록 불정확)

Intensity 특성

- 표면 반사율 의존
- Glass: 거의 반사 없음
- Retroreflective: 포화
- 날씨(비, 안개)에 취약

FAST-LIO 모델

Point uncertainty:
$R = \sigma^2 I_{3\times 3}$ (등방성 가정)

실제: surface 방향에 따라
비등방성이 더 정확

LOAM (Zhang & Singh, RSS 2014)

최초의 실시간 Feature-based LiDAR Odometry

- 논문: “LOAM: Lidar Odometry and Mapping in Real-time”
- Two-thread 구조: Odometry (10Hz) + Mapping (1Hz)
- Edge features + Planar features 추출
- Point-to-edge / Point-to-plane matching

LOAM (Zhang & Singh, RSS 2014)

최초의 실시간 Feature-based LiDAR Odometry

- 논문: “LOAM: Lidar Odometry and Mapping in Real-time”
- Two-thread 구조: Odometry (10Hz) + Mapping (1Hz)
- Edge features + Planar features 추출
- Point-to-edge / Point-to-plane matching

성능

- KITTI 당시 SOTA
- 현재도 baseline
- 64-beam에서 실시간

파생 연구

- LeGO-LOAM (경량화)
- LIO-SAM (IMU 통합)
- FAST-LIO (IEKF 방식)

LOAM 측정 모델 — Point-to-Plane

Residual 정의
Feature point p, 지도 평면: 3점 q_1, q_2, q_3로 정의

$$n = \frac{(q_2 - q_1) \times (q_3 - q_1)}{\|(q_2 - q_1) \times (q_3 - q_1)\|} \quad \text{(normal vector)}$$

$$r = n^\top (Tp - q_1) \quad \text{(point-to-plane distance)}$$

LOAM 측정 모델 — Point-to-Plane

Residual 정의
Feature point p, 지도 평면: 3점 q_1, q_2, q_3로 정의

$$n = \frac{(q_2 - q_1) \times (q_3 - q_1)}{\|(q_2 - q_1) \times (q_3 - q_1)\|} \quad \text{(normal vector)}$$

$$r = n^\top (Tp - q_1) \quad \text{(point-to-plane distance)}$$

Jacobian on $\mathrm{SE}(3)$

$$\frac{\partial r}{\partial [\phi,\, t]} = n^\top \frac{\partial (Rp + t)}{\partial [\phi,\, t]} = n^\top \left[-[Rp]_\times ,\ I \right] \in \mathbb{R}^{1 \times 6}$$

여기서 $[Rp]_\times$는 Rp의 skew-symmetric matrix

LOAM 측정 모델 — Point-to-Plane

Residual 정의
Feature point p, 지도 평면: 3점 q_1, q_2, q_3로 정의

$$n = \frac{(q_2 - q_1) \times (q_3 - q_1)}{\|(q_2 - q_1) \times (q_3 - q_1)\|} \quad \text{(normal vector)}$$

$$r = n^\top (Tp - q_1) \quad \text{(point-to-plane distance)}$$

Jacobian on $\mathrm{SE}(3)$

$$\frac{\partial r}{\partial [\phi,\, t]} = n^\top \frac{\partial (Rp + t)}{\partial [\phi,\, t]} = n^\top \left[-[Rp]_\times ,\ I \right] \in \mathbb{R}^{1 \times 6}$$

여기서 $[Rp]_\times$는 Rp의 skew-symmetric matrix

핵심 관찰
이 구조가 FAST-LIO IEKF의 측정 Jacobian H와 동일한 형태

LeGO-LOAM (Shan & Englot, IROS 2018)

경량화 LOAM — 지상 이동 로봇용

- Ground segmentation: 지면 포인트 분리
- 6DOF → 3DOF 지면 제약 (roll, pitch, z 고정)
- Loop closure: Pose graph optimization
- 실외 매핑에 인기 (주차장, 캠퍼스)

LeGO-LOAM (Shan & Englot, IROS 2018)

경량화 LOAM — 지상 이동 로봇용

- Ground segmentation: 지면 포인트 분리
- 6DOF → 3DOF 지면 제약 (roll, pitch, z 고정)
- Loop closure: Pose graph optimization
- 실외 매핑에 인기 (주차장, 캠퍼스)

Ground Segmentation

- Beam angle 기반
- Ground: 수평 에 가까운 포인트
- Object: 나머지
- 계산 효율 크게 향상

실용 포인트

- Velodyne VLP-16에서 실시간
- CPU 사용량 낮음
- ROS 패키지 공개
- 산업 로봇에 적용 다수

LOAM 계열 한계

주요 한계점

1. **Feature extraction 의존**: featureless 환경(긴 복도, 개활지)에서 실패
2. **고밀도 LiDAR**: 128-beam에서 실시간 어려움
3. **IMU 미통합**: Pure LiDAR $\Rightarrow$ 급격한 운동에 취약
4. **Solid-state 미지원**: 비반복 패턴에 feature extraction 부적합

LOAM 계열 한계

주요 한계점

1. **Feature extraction 의존**: featureless 환경(긴 복도, 개활지)에서 실패
2. **고밀도 LiDAR**: 128-beam에서 실시간 어려움
3. **IMU 미통합**: Pure LiDAR ⇒ 급격한 운동에 취약
4. **Solid-state 미지원**: 비반복 패턴에 feature extraction 부적합

해결 방향

- IMU tightly coupled ⇒ FAST-LIO (IEKF)
- Feature-free ⇒ Direct point-to-plane
- Dynamic map ⇒ ikd-tree (FAST-LIO2)

LIO (LiDAR-Inertial Odometry) 동기

IMU + LiDAR 상호 보완

	IMU	LiDAR
주파수	200Hz	10Hz
지연	낮음	높음
오차	누적 drift	절대 geometry
계산량	매우 낮음	높음

LIO (LiDAR-Inertial Odometry) 동기

IMU + LiDAR 상호 보완

	IMU	LiDAR
주파수	200Hz	10Hz
지연	낮음	높음
오차	누적 drift	절대 geometry
계산량	매우 낮음	높음

Tight Coupling
IMU propagation + LiDAR update를
단일 필터에서 통합 (ESKF)
더 정확, 더 복잡

Loose Coupling
LiDAR odometry + IMU 별도 처리
출력을 pose-level 융합
더 간단, 덜 정확

LIO (LiDAR-Inertial Odometry) 동기

IMU + LiDAR 상호 보완

	IMU	LiDAR
주파수	200Hz	10Hz
지연	낮음	높음
오차	누적 drift	절대 geometry
계산량	매우 낮음	높음

Tight Coupling
IMU propagation + LiDAR update를
단일 필터에서 통합 (ESKF)
더 정확, 더 복잡

Loose Coupling
LiDAR odometry + IMU 별도 처리
출력을 pose-level 융합
더 간단, 덜 정확

FAST-LIO는 Tightly Coupled
IMU propagation + LiDAR IEKF update = 단일 ESKF

LIO-SAM (Shan et al., IROS 2020)

Factor Graph + LiDAR-IMU Tight Coupling

- **IMU preintegration factor**: Forster et al. (2017)
- **LiDAR odometry factor**: scan-to-submap matching
- **GPS factor**: optional absolute position
- **Loop closure factor**: Euclidean distance trigger + ICP

LIO-SAM (Shan et al., IROS 2020)

Factor Graph + LiDAR-IMU Tight Coupling

- **IMU preintegration factor**: Forster et al. (2017)
- **LiDAR odometry factor**: scan-to-submap matching
- **GPS factor**: optional absolute position
- **Loop closure factor**: Euclidean distance trigger + ICP

GTSAM 기반

- iSAM2 (incremental smoothing and mapping)
- 점진적 factor graph update
- Loop closure 시 전체 궤적 재최적화

LIO-SAM (Shan et al., IROS 2020)

Factor Graph + LiDAR-IMU Tight Coupling

- **IMU preintegration factor**: Forster et al. (2017)
- **LiDAR odometry factor**: scan-to-submap matching
- **GPS factor**: optional absolute position
- **Loop closure factor**: Euclidean distance trigger + ICP

GTSAM 기반

- iSAM2 (incremental smoothing and mapping)
- 점진적 factor graph update
- Loop closure 시 전체 궤적 재최적화

Keyframe 기반

이동/회전 임계값 초과 시 keyframe 선택 (≈1–2Hz)
Local sliding window map

LIO-SAM 구조


```
// LIO-SAM 데이터흐름개념도  ()

[IMU 200Hz]
    |
    v
IMU Preintegration Factor
    |
    +---> [GTSAM Factor Graph] <---+
    |                              |
[LiDAR 10Hz]               [GPS (optional)]
    |                              |
    v                              |
Scan-to-Submap Matching           GPS Factor
    |
LiDAR Odometry Factor -------->  [GTSAM Graph]
                                       |
[Loop Closure Detector]                v
  SC-descriptor + ICP -----> Loop Closure Factor
                                       |
                                iSAM2 Optimization
                                       |
                                Corrected Poses + Map
```

Keyframe 선택
이동 >1 m 또는 회전 $>10°$ $\Rightarrow$ new keyframe
Local map: 최근 N개 keyframe point cloud 합산

LIO-SAM 한계

실용적 한계

1. **계산량**: GTSAM + ICP 동시 실행 ⇒ 높은 CPU 요구
2. **Keyframe latency**: full LiDAR rate로 실시간 불가
3. **Solid-state 미지원**: 비반복 스캔 패턴에 ICP 불안정
4. **Feature extraction**: LOAM 스타일 feature 의존

LIO-SAM 한계

실용적 한계

1. **계산량**: GTSAM + ICP 동시 실행 ⇒ 높은 CPU 요구
2. **Keyframe latency**: full LiDAR rate로 실시간 불가
3. **Solid-state 미지원**: 비반복 스캔 패턴에 ICP 불안정
4. **Feature extraction**: LOAM 스타일 feature 의존

FAST-LIO의 해결책

- Feature extraction 제거 ⇒ direct point-to-plane
- IEKF ⇒ 경량, 실시간 (10Hz at full LiDAR rate)
- Solid-state LiDAR 완전 지원
- ikd-tree ⇒ 동적 map 관리 (FAST-LIO2)

FAST-LIO 개요 (He et al., RA-L 2021)

논문
“FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter”
IEEE Robotics and Automation Letters, 2021

FAST-LIO 개요 (He et al., RA-L 2021)

논문
“FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter”
IEEE Robotics and Automation Letters, 2021

핵심 기여

1. **IEKF on manifold**: SO(3)에서 iterated EKF
2. **Direct point-to-plane**: feature extraction 불필요
3. **Fast nearest neighbor**: kd-tree (후속: ikd-tree)
4. **Per-point undistortion**: IEKF 내부 통합

FAST-LIO 개요 (He et al., RA-L 2021)

논문
“FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter”
IEEE Robotics and Automation Letters, 2021

핵심 기여

1. **IEKF on manifold**: SO(3)에서 iterated EKF
2. **Direct point-to-plane**: feature extraction 불필요
3. **Fast nearest neighbor**: kd-tree (후속: ikd-tree)
4. **Per-point undistortion**: IEKF 내부 통합

주요 결과

- MaiCity: ATE 0.2 m / 2 km
- 실시간: Intel i7, 64-beam LiDAR, 10Hz
- GitHub: `hku-mars/FAST_LIO` (GPL-2.0)

IEKF — Iterated Extended Kalman Filter

표준 EKF vs IEKF

- **EKF**: 한 번의 linearization ⇒ 비선형 오차 큼
- **IEKF**: 반복적 linearization + update (Gauss-Newton style)

IEKF — Iterated Extended Kalman Filter

표준 EKF vs IEKF

- **EKF**: 한 번의 linearization ⇒ 비선형 오차 큼
- **IEKF**: 반복적 linearization + update (Gauss-Newton style)

IEKF iteration l (FAST-LIO2 Eq. 18, 표준 부호 규약)

표기: $\tilde{x}^l = \hat{x}^l \boxminus \hat{x}^-$ (prior로부터의 편차), $r^l = z - h(\hat{x}^l)$ (innovation).

$$\hat{x}^{l+1} = \hat{x}^l \boxplus \left(K^l r^l \; - \; (I - K^l H^l)\, \tilde{x}^l \right)$$

$$K^l = P^- H^{l\top} (H^l P^- H^{l\top} + R)^{-1}$$

- $l = 0$: $\tilde{x}^0 = \mathbf{0} \Rightarrow \hat{x}^1 = \hat{x}^- \boxplus K^0 r^0$ (표준 EKF)
- $-(I - KH)\tilde{x}^l$ 항: 재선형화 보정 (Bell&Cathey 1993의 GN step)
- 수렴 기준: $\|\hat{x}^{l+1} \boxminus \hat{x}^l\|_\infty < \epsilon$

IEKF — Iterated Extended Kalman Filter

표준 EKF vs IEKF

- **EKF**: 한 번의 linearization ⇒ 비선형 오차 큼
- **IEKF**: 반복적 linearization + update (Gauss-Newton style)

IEKF iteration l (FAST-LIO2 Eq. 18, 표준 부호 규약)

표기: $\tilde{x}^l = \hat{x}^l \boxminus \hat{x}^-$ (prior로부터의 편차), $r^l = z - h(\hat{x}^l)$ (innovation).

$$\hat{x}^{l+1} = \hat{x}^l \boxplus \left(K^l r^l \; - \; (I - K^l H^l)\, \tilde{x}^l \right)$$

$$K^l = P^- H^{l\top} (H^l P^- H^{l\top} + R)^{-1}$$

- $l = 0$: $\tilde{x}^0 = \mathbf{0} \Rightarrow \hat{x}^1 = \hat{x}^- \boxplus K^0 r^0$ (표준 EKF)
- $-(I - KH)\tilde{x}^l$ 항: 재선형화 보정 (Bell&Cathey 1993의 GN step)
- 수렴 기준: $\|\hat{x}^{l+1} \boxminus \hat{x}^l\|_\infty < \epsilon$

왜 IEKF인가?

Point-to-plane은 강한 비선형 관측 모델

IEKF가 EKF 대비 수렴 정확도 현저히 향상 (= Gauss-Newton 수렴)

상태 벡터 (15D)

$$x = [\underbrace{{}^{W}_{I}R}_{\mathrm{SO}(3)}, \underbrace{{}^{W}p_I}_{\mathbb{R}^3}, \underbrace{{}^{W}v_I}_{\mathbb{R}^3}, \underbrace{b_g}_{\mathbb{R}^3}, \underbrace{b_a}_{\mathbb{R}^3}]$$

FAST-LIO IEKF on Manifold

상태 벡터 (15D)

$$x = [\underbrace{{}^{W}_{I}R}_{\mathrm{SO}(3)},\ \underbrace{{}^{W}p_I}_{\mathbb{R}^3},\ \underbrace{{}^{W}v_I}_{\mathbb{R}^3},\ \underbrace{b_g}_{\mathbb{R}^3},\ \underbrace{b_a}_{\mathbb{R}^3}]$$

Error state (15D)

$$\delta x = [\delta\phi,\ \delta p,\ \delta v,\ \delta b_g,\ \delta b_a] \in \mathbb{R}^{15}$$

FAST-LIO IEKF on Manifold

상태 벡터 (15D)

$$x = [\underbrace{{}^W_I R}_{\mathrm{SO}(3)},\ \underbrace{{}^W p_I}_{\mathbb{R}^3},\ \underbrace{{}^W v_I}_{\mathbb{R}^3},\ \underbrace{b_g}_{\mathbb{R}^3},\ \underbrace{b_a}_{\mathbb{R}^3}]$$

Error state (15D)

$$\delta x = [\delta\phi,\ \delta p,\ \delta v,\ \delta b_g,\ \delta b_a] \in \mathbb{R}^{15}$$

Manifold Update ($\boxplus$)

$$\hat{x} \leftarrow \hat{x} \boxplus \delta x^* \quad \text{where}$$

$$\hat{R} \leftarrow \hat{R}\,\mathrm{Exp}(\delta\phi), \quad \hat{p} \leftarrow \hat{p} + \delta p, \quad \hat{v} \leftarrow \hat{v} + \delta v$$

IEKF iteration: linearize → solve → retract (반복)

FAST-LIO 측정 모델 — Point-to-Plane

LiDAR 포인트 → World frame 변환

$$p_k^W = {}_I^W R \cdot {}_L^I R \cdot p_k^L + {}^W p_I + {}_I^W R \cdot {}^I p_L$$

- ${}_L^I R, {}^I p_L$: LiDAR-IMU extrinsic (사전 캘리브레이션)
- p_k^L: LiDAR frame의 포인트

FAST-LIO 측정 모델 — Point-to-Plane

LiDAR 포인트 → World frame 변환

$$p_k^W = {}_I^W R \cdot {}_L^I R \cdot p_k^L + {}^W p_I + {}_I^W R \cdot {}^I p_L$$

- ${}_L^I R, {}^I p_L$: LiDAR-IMU extrinsic (사전 캘리브레이션)
- p_k^L: LiDAR frame의 포인트

Plane fitting + Residual
ikd-tree에서 $k = 5$ nearest neighbors ⇒ PCA로 평면 피팅

$$r_k = n^\top (p_k^W - \bar{q}) \quad \text{(point-to-plane signed distance)}$$

n: 평면 법선벡터, $\bar{q}$: 평면 centroid

FAST-LIO 측정 모델 — Point-to-Plane

LiDAR 포인트 → World frame 변환

$$p_k^W = {}_I^W R \cdot {}_L^I R \cdot p_k^L + {}^W p_I + {}_I^W R \cdot {}^I p_L$$

- ${}_L^I R, {}^I p_L$: LiDAR-IMU extrinsic (사전 캘리브레이션)
- p_k^L: LiDAR frame의 포인트

Plane fitting + Residual
ikd-tree에서 $k = 5$ nearest neighbors ⇒ PCA로 평면 피팅

$$r_k = n^\top (p_k^W - \bar{q}) \quad \text{(point-to-plane signed distance)}$$

n: 평면 법선벡터, $\bar{q}$: 평면 centroid

유효성 검사
평면 fitting 품질: residual이 임계값 이내인 경우만 사용
$r_k > d_{\max}$ 이면 해당 포인트 제외 (outlier rejection)

Point-to-Plane Jacobian 유도

H_k **유도**
$r_k = n^\top p_k^W(x)$이므로:

$$H_k = n^\top \frac{\partial p_k^W}{\partial \delta x} \in \mathbb{R}^{1\times 15}$$

각 블록별 편미분:

$$\frac{\partial p_k^W}{\partial \delta \phi} = -\left[{}_I^W R \cdot {}_L^I R \cdot p_k^L\right]_\times \quad \in \mathbb{R}^{3\times 3}$$

$$\frac{\partial p_k^W}{\partial \delta p} = I_3$$

$$\frac{\partial p_k^W}{\partial \delta v},\ \frac{\partial p_k^W}{\partial \delta b_g},\ \frac{\partial p_k^W}{\partial \delta b_a} = 0$$

Point-to-Plane Jacobian 유도

H_k 유도
$r_k = n^\top p_k^W(x)$이므로:

$$H_k = n^\top \frac{\partial p_k^W}{\partial \delta x} \in \mathbb{R}^{1\times 15}$$

각 블록별 편미분:

$$\frac{\partial p_k^W}{\partial \delta \phi} = -\left[{}_I^W R \cdot {}_L^I R \cdot p_k^L\right]_\times \quad \in \mathbb{R}^{3\times 3}$$

$$\frac{\partial p_k^W}{\partial \delta p} = I_3$$

$$\frac{\partial p_k^W}{\partial \delta v},\ \frac{\partial p_k^W}{\partial \delta b_g},\ \frac{\partial p_k^W}{\partial \delta b_a} = 0$$

Full Jacobian

$$H_k = n^\top \left[-\left[{}_I^W R\, {}_L^I R\, p_k^L\right]_\times,\ I_3,\ 0_{3\times 3},\ 0_{3\times 3},\ 0_{3\times 3}\right] \in \mathbb{R}^{1\times 15}$$

N개 포인트: $H \in \mathbb{R}^{N\times 15}$ (매우 긴 행렬)

ESKF Propagation (L5 복습)
이산화 행렬 F_d, G_d 사용:

$$P^- \leftarrow F_d \, P \, F_d^\top + G_d \, Q_c \, G_d^\top$$

15D error state의 F, G 행렬: 5강 ESKF와 동일 구조

FAST-LIO 공분산 전파 — IMU Propagation

ESKF Propagation (L5 복습)
이산화 행렬 F_d, G_d 사용:

$$P^- \leftarrow F_d \, P \, F_d^\top + G_d \, Q_c \, G_d^\top$$

15D error state의 F, G 행렬: 5강 ESKF와 동일 구조

Per-point Undistortion 통합

- IMU 적분 시 각 포인트 timestamp에서 자세 기록
- IEKF iteration 시작 전 undistortion 완료
- 별도 전처리 단계 불필요 → latency 감소

FAST-LIO 공분산 전파 — IMU Propagation

ESKF Propagation (L5 복습)
이산화 행렬 F_d, G_d 사용:

$$P^- \leftarrow F_d \, P \, F_d^\top + G_d \, Q_c \, G_d^\top$$

15D error state의 F, G 행렬: 5강 ESKF와 동일 구조

Per-point Undistortion 통합

- IMU 적분 시 각 포인트 timestamp에서 자세 기록
- IEKF iteration 시작 전 undistortion 완료
- 별도 전처리 단계 불필요 → latency 감소

FAST-LIO 특이점
Per-point undistortion을 IEKF 내부에 통합
IMU propagation 중 자세 보간 수행 → 효율적

FAST-LIO IEKF 알고리즘 전체

입력: IMU measurements + LiDAR scan
Step 1: IMU Propagation

$$\hat{x}^{(0)}, P^{-} \leftarrow \text{IMU propagation} + \text{undistortion}$$

Step 2: LiDAR Point Residuals
각 포인트 p_k: ikd-tree에서 nearest neighbors → plane fitting → r_k, H_k

Step 3: IEKF Update at iteration l (with $\tilde{x}^l = \hat{x}^l \boxminus \hat{x}^{(0)}$, $r^l = z - h(\hat{x}^l)$)

$$\begin{aligned} K^l &= P^{-} H^{l\top} \left(H^l P^{-} H^{l\top} + R \right)^{-1} \\ \delta x^* &= K^l r^l \; - \; \left(I - K^l H^l \right) \tilde{x}^l \\ \hat{x}^{l+1} &\leftarrow \hat{x}^l \boxplus \delta x^* \end{aligned}$$

Step 4: 수렴 확인 $\|\delta x^*\|_\infty < \epsilon$이면 종료, 아니면 Step 2로

Step 5: 공분산 업데이트 (Joseph form 권장) + Map 업데이트

$$P^{+} = (I - KH) P^{-} (I - KH)^{\top} + K R K^{\top}$$

ikd-tree ← Add new points

FAST-LIO 초기 버전: PCL kd-tree

- Map: 3D kd-tree (`pcl::KdTreeFLANN`)
- 각 query point: $k = 5$ nearest neighbors
- 평면 fitting: 5점 PCA $\rightarrow$ normal vector

kd-tree 기반 Nearest Neighbor Search

FAST-LIO 초기 버전: PCL kd-tree

- Map: 3D kd-tree (`pcl::KdTreeFLANN`)
- 각 query point: $k = 5$ nearest neighbors
- 평면 fitting: 5점 PCA $\rightarrow$ normal vector

복잡도 분석

- Query: $O(\log N)$ per point
- Tree build: $O(N \log N)$
- N개 포인트 query: $O(N \log N)$

kd-tree 기반 Nearest Neighbor Search

FAST-LIO 초기 버전: PCL kd-tree

- Map: 3D kd-tree (`pcl::KdTreeFLANN`)
- 각 query point: $k = 5$ nearest neighbors
- 평면 fitting: 5점 PCA $\rightarrow$ normal vector

복잡도 분석

- Query: $O(\log N)$ per point
- Tree build: $O(N \log N)$
- N개 포인트 query: $O(N \log N)$

핵심 한계
Dynamic insert: 매 scan마다 tree rebuild $\rightarrow O(N \log N)$
Map이 커질수록 점점 느려짐
$\Rightarrow$ FAST-LIO2에서 ikd-tree로 해결

FAST-LIO 성능

벤치마크 결과

- MaiCity dataset: ATE 0.2 m / 2 km (실외 밤)
- Hilti SLAM Challenge: competitive
- 실시간: Intel i7, 64-beam, 10Hz 달성

FAST-LIO 성능

벤치마크 결과

- MaiCity dataset: ATE 0.2 m / 2 km (실외 밤)
- Hilti SLAM Challenge: competitive
- 실시간: Intel i7, 64-beam, 10Hz 달성

강인성

- Degeneracy handling
- $H^\top H$ 최솟값 eigenvalue 체크
- 취약 방향 covariance 확대
- 터널, 복도에서도 동작

오픈소스

`github.com/hku-mars/FAST_LIO`
GPL-2.0 라이선스
ROS 1/2 모두 지원
Velodyne, Ouster, Livox

FAST-LIO 한계

1. **kd-tree**: dynamic map update 비용 높음 ($O(N \log N)$ rebuild)
2. **Loop closure 없음**: open-loop drift (장거리)
3. **메모리**: map 제한 없이 증가
4. **Downsampling**: 별도 filter 필요

FAST-LIO 한계 및 FAST-LIO2 동기

FAST-LIO 한계

1. **kd-tree**: dynamic map update 비용 높음 ($O(N \log N)$ rebuild)
2. **Loop closure 없음**: open-loop drift (장거리)
3. **메모리**: map 제한 없이 증가
4. **Downsampling**: 별도 filter 필요

FAST-LIO2 핵심 개선

- **ikd-tree**: Incremental kd-tree
  - Insert/delete: $O(\log N)$ amortized
  - Dynamic map management
  - Built-in downsampling
- 결과: 2–3배 빠른 map update, 메모리 효율

FAST-LIO2 (Xu et al., TRO 2022)

논문
“FAST-LIO2: Fast Direct LiDAR-Inertial Odometry”
IEEE Transactions on Robotics, 2022

FAST-LIO2 (Xu et al., TRO 2022)

논문
"FAST-LIO2: Fast Direct LiDAR-Inertial Odometry"
IEEE Transactions on Robotics, 2022

주요 기여

1. **ikd-tree**: Incremental kd-tree 도입
   - Point insertion/deletion: $O(\log N)$
   - Dynamic map: 새 포인트 추가, 오래된 포인트 삭제
   - Downsampling: ikd-tree 내부에 통합
2. **Direct**: feature extraction 없이 raw points 직접 사용
3. **IEKF**: FAST-LIO와 동일 구조, map 부분만 개선

FAST-LIO2 (Xu et al., TRO 2022)

논문
"FAST-LIO2: Fast Direct LiDAR-Inertial Odometry"
IEEE Transactions on Robotics, 2022

주요 기여

1. **ikd-tree**: Incremental kd-tree 도입
   - Point insertion/deletion: $O(\log N)$
   - Dynamic map: 새 포인트 추가, 오래된 포인트 삭제
   - Downsampling: ikd-tree 내부에 통합
2. **Direct**: feature extraction 없이 raw points 직접 사용
3. **IEKF**: FAST-LIO와 동일 구조, map 부분만 개선

결과
FAST-LIO 대비 동일 정확도 + 더 빠른 map update
`github.com/hku-mars/FAST_LIO` (동일 저장소, v2)

ikd-Tree 동기 — 왜 새로운 자료구조인가

기존 kd-tree 문제

- PCL kd-tree: **static** 구조
- 포인트 추가 시: 전체 tree rebuild $O(N \log N)$
- LiDAR SLAM: 매 scan마다 수천 포인트 추가
- Map 크기 $\uparrow \Rightarrow$ rebuild 시간 $\uparrow \Rightarrow$ 실시간 불가

ikd-Tree 동기 — 왜 새로운 자료구조인가

기존 kd-tree 문제

- PCL kd-tree: **static** 구조
- 포인트 추가 시: 전체 tree rebuild $O(N \log N)$
- LiDAR SLAM: 매 scan마다 수천 포인트 추가
- Map 크기 $\uparrow \Rightarrow$ rebuild 시간 $\uparrow \Rightarrow$ 실시간 불가

ikd-tree 해결책

- He et al., “ikd-Tree: An Incremental K-D Tree for Robotic Applications”
- IEEE Robotics and Automation Letters, 2022
- Incremental update: $O(\log N)$ per insertion (amortized)
- Lazy deletion, partial rebuild

ikd-Tree 동기 — 왜 새로운 자료구조인가

기존 kd-tree 문제

- PCL kd-tree: **static** 구조
- 포인트 추가 시: 전체 tree rebuild $O(N \log N)$
- LiDAR SLAM: 매 scan마다 수천 포인트 추가
- Map 크기 $\uparrow \Rightarrow$ rebuild 시간 $\uparrow \Rightarrow$ 실시간 불가

ikd-tree 해결책

- He et al., “ikd-Tree: An Incremental K-D Tree for Robotic Applications”
- IEEE Robotics and Automation Letters, 2022
- Incremental update: $O(\log N)$ per insertion (amortized)
- Lazy deletion, partial rebuild

실용 효과
map update 시간: kd-tree 대비 2–3배 단축
FAST-LIO2의 핵심 기여

ikd-Tree 구조 — 균형 이진 트리

기본 구조

- Balanced BST on 3D space
- 분할 기준: 레벨별 축 교대 (x, y, z, x, ...)
- 각 노드: 3D 포인트 + 자식 포인터 + 삭제 플래그

ikd-Tree 구조 — 균형 이진 트리

기본 구조

- Balanced BST on 3D space
- 분할 기준: 레벨별 축 교대 (x, y, z, x, ...)
- 각 노드: 3D 포인트 + 자식 포인터 + 삭제 플래그

균형 유지 전략

- α-balanced condition: $|L| \leq \alpha|T|$, $|R| \leq \alpha|T|$
- 불균형 감지 시: 해당 subtree만 partial rebuild
- $\alpha = 0.625$ (FAST-LIO2 default)

ikd-Tree 구조 — 균형 이진 트리

기본 구조

- Balanced BST on 3D space
- 분할 기준: 레벨별 축 교대 (x, y, z, x, ...)
- 각 노드: 3D 포인트 + 자식 포인터 + 삭제 플래그

균형 유지 전략

- α-balanced condition: $|L| \leq \alpha|T|$, $|R| \leq \alpha|T|$
- 불균형 감지 시: 해당 subtree만 partial rebuild
- $\alpha = 0.625$ (FAST-LIO2 default)

삭제: Lazy deletion
물리적 삭제 대신 flag 설정
실제 삭제는 rebuild 시

크기 정보
각 노드: subtree 크기 $|T|$ 저장
유효 노드 수 $|T_{valid}|$ 별도 관리

ikd-Tree 삽입 알고리즘

삽입 절차
새 포인트 p 삽입:

1. 분할 기준에 따라 트리 탐색 (leaf까지)
2. Leaf에 새 노드 삽입
3. 부모 노드들의 subtree 크기 갱신
4. Balance check: α-balanced 조건 위반 시 subtree rebuild

ikd-Tree 삽입 알고리즘

삽입 절차
새 포인트 p 삽입:

1. 분할 기준에 따라 트리 탐색 (leaf까지)
2. Leaf에 새 노드 삽입
3. 부모 노드들의 subtree 크기 갱신
4. Balance check: α-balanced 조건 위반 시 subtree rebuild

Balance 조건

$$|L| \leq \alpha|T|, \quad |R| \leq \alpha|T|,$$
$$|T_{invalid}| \leq \beta|T|$$

위반 시: 해당 subtree 수집 $\rightarrow$ 균형 재구성
전체 tree rebuild 아님 $\Rightarrow$ Amortized $O(\log N)$

ikd-Tree 삽입 알고리즘

삽입 절차
새 포인트 p 삽입:

1. 분할 기준에 따라 트리 탐색 (leaf까지)
2. Leaf에 새 노드 삽입
3. 부모 노드들의 subtree 크기 갱신
4. Balance check: α-balanced 조건 위반 시 subtree rebuild

Balance 조건

$$|L| \leq \alpha|T|, \quad |R| \leq \alpha|T|,$$
$$|T_{invalid}| \leq \beta|T|$$

위반 시: 해당 subtree 수집 $\rightarrow$ 균형 재구성
전체 tree rebuild 아님 $\Rightarrow$ Amortized $O(\log N)$

Bulk insertion
여러 포인트 동시 삽입: $O(M \log N)$
FAST-LIO2: 매 scan 후 batch insert

ikd-Tree 삭제

Lazy Deletion

- 노드에 `deleted` flag 설정
- 실제 물리적 제거 없음
- 쿼리 시: deleted node는 결과에서 제외
- Rebuild 시: 물리적 제거 수행

ikd-Tree 삭제

Lazy Deletion

- 노드에 `deleted` flag 설정
- 실제 물리적 제거 없음
- 쿼리 시: deleted node는 결과에서 제외
- Rebuild 시: 물리적 제거 수행

FAST-LIO2에서 삭제 활용

- **Downsampling**: voxel 내 포인트 수 초과 시 초과분 삭제
- **Box delete**: 로봇 이동으로 멀어진 영역의 포인트 삭제
  - Bounding box 지정 $\rightarrow$ 내부 모든 포인트 lazy delete
  - $O(\log N + k)$, k: 삭제 포인트 수

ikd-Tree 삭제

Lazy Deletion

- 노드에 `deleted` flag 설정
- 실제 물리적 제거 없음
- 쿼리 시: deleted node는 결과에서 제외
- Rebuild 시: 물리적 제거 수행

FAST-LIO2에서 삭제 활용

- **Downsampling**: voxel 내 포인트 수 초과 시 초과분 삭제
- **Box delete**: 로봇 이동으로 멀어진 영역의 포인트 삭제
  - Bounding box 지정 $\rightarrow$ 내부 모든 포인트 lazy delete
  - $O(\log N + k)$, k: 삭제 포인트 수

실용 설정
Map 크기 제한: `cube_len = 1000m`
로봇 위치에서 500m 이상 벗어난 영역 주기적 삭제

ikd-Tree 쿼리 — Nearest Neighbor

KNN Search

1. Query point q로 트리 탐색
2. Priority queue (max-heap): k개 최근접 후보 유지
3. Pruning: 현재 best k-th distance보다 멀면 subtree skip
4. Deleted nodes: 결과에서 자동 제외

복잡도: $O(\log N)$ amortized

ikd-Tree 쿼리 — Nearest Neighbor

KNN Search

1. Query point q로 트리 탐색
2. Priority queue (max-heap): k개 최근접 후보 유지
3. Pruning: 현재 best k-th distance보다 멀면 subtree skip
4. Deleted nodes: 결과에서 자동 제외

복잡도: $O(\log N)$ amortized

추가 쿼리 종류

- **Range search**: 반경 r 이내 모든 포인트
- **Box search**: 3D bounding box 내 포인트
- **Count**: 영역 내 포인트 수 ($O(\log N)$)

ikd-Tree 쿼리 — Nearest Neighbor

KNN Search

1. Query point q로 트리 탐색
2. Priority queue (max-heap): k개 최근접 후보 유지
3. Pruning: 현재 best k-th distance보다 멀면 subtree skip
4. Deleted nodes: 결과에서 자동 제외

복잡도: $O(\log N)$ amortized

추가 쿼리 종류

- **Range search**: 반경 r 이내 모든 포인트
- **Box search**: 3D bounding box 내 포인트
- **Count**: 영역 내 포인트 수 ($O(\log N)$)

FAST-LIO2 사용
각 LiDAR 포인트마다 $k = 5$ KNN $\rightarrow$ plane fitting
전체 scan: $O(N \log N)$

ikd-Tree vs 다른 동적 구조

자료구조 성능 비교

연산	ikd-tree	Octree	Hash voxel	Static kd-tree
Insert (1점)	$O(\log N)^*$	$O(\log N)$	$O(1)$	$O(N \log N)$ rebuild
KNN	$O(\log N)$	$O(\log N)$	N/A	$O(\log N)$
Range search	$O(\log N)$	$O(\log N)$	$O(1)$ 각 voxel	$O(\log N)$
Box delete	$O(\log N)$	$O(\log N)$	$O(1)$ per voxel	$O(N)$ scan
메모리	$O(N)$	$O(N)$	$O(N/v)$	$O(N)$

*amortized

ikd-Tree vs 다른 동적 구조

자료구조 성능 비교

연산	ikd-tree	Octree	Hash voxel	Static kd-tree
Insert (1점)	$O(\log N)^*$	$O(\log N)$	$O(1)$	$O(N \log N)$ rebuild
KNN	$O(\log N)$	$O(\log N)$	N/A	$O(\log N)$
Range search	$O(\log N)$	$O(\log N)$	$O(1)$ 각 voxel	$O(\log N)$
Box delete	$O(\log N)$	$O(\log N)$	$O(1)$ per voxel	$O(N)$ scan
메모리	$O(N)$	$O(N)$	$O(N/v)$	$O(N)$

*amortized

FAST-LIO2 선택 이유

ikd-tree: KNN 정확도 + 동적 update 모두 우수

Hash voxel (iVox): insert 더 빠르지만 KNN 근사치

⇒ 정확도가 중요한 IEKF에는 ikd-tree 적합

FAST-LIO2 Direct Point Matching

Feature-free 접근

1. Undistorted raw points를 world frame으로 변환
2. 각 포인트 $\rightarrow$ ikd-tree KNN ($k = 5$)
3. 5 neighbors로 평면 fitting (PCA)
4. Point-to-plane residual r_k + Jacobian H_k 계산
5. IEKF update

FAST-LIO2 Direct Point Matching

Feature-free 접근

1. Undistorted raw points를 world frame으로 변환
2. 각 포인트 $\rightarrow$ ikd-tree KNN ($k = 5$)
3. 5 neighbors로 평면 fitting (PCA)
4. Point-to-plane residual r_k + Jacobian H_k 계산
5. IEKF update

FAST-LIO2 vs FAST-LIO

- kd-tree $\rightarrow$ **ikd-tree**
- IEKF 구조: 동일
- Feature: 동일 (없음)
- 차이: map 관리

Batch 처리
전체 scan 포인트를 한번에
Parallel computation 가능
GPU 가속 연구 진행 중

FAST-LIO2 Map 관리

Map Update 흐름
매 scan 후:

1. **New points**: IEKF update 완료 후 ikd-tree에 삽입
2. **Downsampling**: ikd-tree 내부 voxel별 포인트 수 제한
3. **Old points**: 로봇 위치에서 먼 영역 box delete

FAST-LIO2 Map 관리

Map Update 흐름

매 scan 후:

1. **New points**: IEKF update 완료 후 ikd-tree에 삽입
2. **Downsampling**: ikd-tree 내부 voxel별 포인트 수 제한
3. **Old points**: 로봇 위치에서 먼 영역 box delete

Map 파라미터

- `filter_size_surf`: map voxel size (default 0.5 m)
- `cube_len`: local map 크기 (default 1000 m)
- `det_range`: 유효 LiDAR 탐지 거리

FAST-LIO2 Map 관리

Map Update 흐름
매 scan 후:

1. **New points**: IEKF update 완료 후 ikd-tree에 삽입
2. **Downsampling**: ikd-tree 내부 voxel별 포인트 수 제한
3. **Old points**: 로봇 위치에서 먼 영역 box delete

Map 파라미터

- `filter_size_surf`: map voxel size (default 0.5 m)
- `cube_len`: local map 크기 (default 1000 m)
- `det_range`: 유효 LiDAR 탐지 거리

주의: Loop Closure 없음
FAST-LIO2는 오직 odometry
장거리 이동 시 drift 누적 (open-loop)
Loop closure가 필요하면: SLAM 모듈 별도 추가

iVox — 대안적 동적 자료구조

iVox (Bai et al., Faster-LIO, 2022)

- Voxel-based spatial hashing
- 각 voxel: 포인트 리스트 저장
- Insert: $O(1)$ (hash table lookup)
- KNN: voxel 내 선형 탐색 + 주변 voxel 체크

iVox — 대안적 동적 자료구조

iVox (Bai et al., Faster-LIO, 2022)

- Voxel-based spatial hashing
- 각 voxel: 포인트 리스트 저장
- Insert: $O(1)$ (hash table lookup)
- KNN: voxel 내 선형 탐색 + 주변 voxel 체크

iVox 장점

- Insert 매우 빠름
- 구현 단순
- Cache-friendly
- 고주파 LiDAR에 유리

iVox 단점

- KNN: 근사치 (voxel-limited)
- Voxel 경계 포인트: 탐색 불정확
- 파라미터 민감도 높음

iVox — 대안적 동적 자료구조

iVox (Bai et al., Faster-LIO, 2022)

- Voxel-based spatial hashing
- 각 voxel: 포인트 리스트 저장
- Insert: $O(1)$ (hash table lookup)
- KNN: voxel 내 선형 탐색 + 주변 voxel 체크

iVox 장점

- Insert 매우 빠름
- 구현 단순
- Cache-friendly
- 고주파 LiDAR에 유리

iVox 단점

- KNN: 근사치 (voxel-limited)
- Voxel 경계 포인트: 탐색 불정확
- 파라미터 민감도 높음

트렌드
초고속 LiDAR (256-beam, 20Hz): iVox 선호
정확도 우선: ikd-tree 유지

FAST-LIO2 성능 비교

공개 데이터셋 결과

데이터셋	ATE	비고
NTU VIRAL (UAV 실내)	≈ 0.1 m	실시간
Hilti 2022 SLAM	Top tier	건설 환경
M1600 (16-beam)	실시간	저가 LiDAR
Livox Avia	완전 지원	Solid-state

FAST-LIO2 성능 비교

공개 데이터셋 결과

데이터셋	ATE	비고
NTU VIRAL (UAV 실내)	≈ 0.1 m	실시간
Hilti 2022 SLAM	Top tier	건설 환경
M1600 (16-beam)	실시간	저가 LiDAR
Livox Avia	완전 지원	Solid-state

실용 성능

- Intel i7 기준: 64-beam 10Hz 실시간
- Livox Avia: 200Hz IMU + 10Hz scan, 실시간
- Memory: map 크기에 따라 0.5–2 GB

FAST-LIO2 성능 비교

공개 데이터셋 결과

데이터셋	ATE	비고
NTU VIRAL (UAV 실내)	≈ 0.1 m	실시간
Hilti 2022 SLAM	Top tier	건설 환경
M1600 (16-beam)	실시간	저가 LiDAR
Livox Avia	완전 지원	Solid-state

실용 성능

- Intel i7 기준: 64-beam 10Hz 실시간
- Livox Avia: 200Hz IMU + 10Hz scan, 실시간
- Memory: map 크기에 따라 0.5–2 GB

오픈소스
`github.com/hku-mars/FAST_LIO` (GPL-2.0)
ROS1 Noetic / ROS2 Humble 지원

Degeneracy Detection in LiDAR

Degenerate 환경

- 긴 복도: 한 방향으로만 평면 제약 $\Rightarrow$ 이동 방향 관측 불가
- 개활지: 수직 제약 없음 $\Rightarrow$ Z 방향 불관측
- 터널: 모든 포인트가 원통형 $\Rightarrow$ 축 방향 모호

Degeneracy Detection in LiDAR

Degenerate 환경

- 긴 복도: 한 방향으로만 평면 제약 $\Rightarrow$ 이동 방향 관측 불가
- 개활지: 수직 제약 없음 $\Rightarrow$ Z 방향 불관측
- 터널: 모든 포인트가 원통형 $\Rightarrow$ 축 방향 모호

검출 및 처리

$$H^\top H = U\,\Lambda\,U^\top, \quad \lambda_{\text{min}} < \lambda_{\text{thresh}}$$

1. λ_{min}이 임계값 미만: 해당 방향이 취약
2. 취약 방향 covariance 확대 ($\Rightarrow$ LiDAR 정보 무시)
3. IMU 정보에 더 의존

Degeneracy Detection in LiDAR

Degenerate 환경

- 긴 복도: 한 방향으로만 평면 제약 $\Rightarrow$ 이동 방향 관측 불가
- 개활지: 수직 제약 없음 $\Rightarrow$ Z 방향 불관측
- 터널: 모든 포인트가 원통형 $\Rightarrow$ 축 방향 모호

검출 및 처리

$$H^\top H = U\,\Lambda\,U^\top, \quad \lambda_{\text{min}} < \lambda_{\text{thresh}}$$

1. λ_{min}이 임계값 미만: 해당 방향이 취약
2. 취약 방향 covariance 확대 ($\Rightarrow$ LiDAR 정보 무시)
3. IMU 정보에 더 의존

FAST-LIO2 구현
`detect_lie_group_degen()` 함수
eigenvalue 체크 + 방향별 gain 조정

FAST-LIO2 코드 구조


```
// laserMapping.cpp (IEKF 메인루프 , 단순화)
for (int iter = 0; iter < NUM_MAX_ITERATIONS; iter++) {
  // 1. World 으로frame 변환 (undistortion 포함)
  pointBodyToWorld(feats_down_body, feats_down_world, kf.get_x());

  // 2. ikd-tree nearest neighbor search
  for (int i = 0; i < feats_down_size; i++) {
    ikdtree.Nearest_Search(feats_down_world->points[i],
                           NUM_MATCH_POINTS, neighbors, dist);
    // 3. 평면 fitting, residual 계산
    if (esti_plane(pabcd, neighbors, 0.1f)) { // pabcd = [n; d]
      float res = pabcd(0)*p.x + pabcd(1)*p.y
                + pabcd(2)*p.z + pabcd(3);
      if (abs(res) > THRESHOLD) continue; // outlier 제거
      res_last[i] = res;
      // H_i 계산 (Jacobian)
      H.row(i) = compute_H(pabcd, feats_down_world->points[i],
                           kf.get_x());
    }
  }
  // 4. IEKF update (ESEKF on manifold)
  kf.update_iterated_dyn_share_modified(res_last, H, R_lidar);
  if (kf.converged()) break;
}
// 5. Map 업데이트 (ikd-tree 삽입)
ikdtree.Add_Points(feats_down_world->points, true);
```

FAST-LIVO 개요 (Zheng et al., RA-L 2022)

논문
“FAST-LIVO: Fast and Tightly-coupled Sparse-Direct LiDAR-Inertial-Visual Odometry”
IEEE Robotics and Automation Letters, 2022

FAST-LIVO 개요 (Zheng et al., RA-L 2022)

논문
“FAST-LIVO: Fast and Tightly-coupled Sparse-Direct LiDAR-Inertial-Visual Odometry”
IEEE Robotics and Automation Letters, 2022

핵심 기여

- **3-sensor tight coupling**: LiDAR + IMU + Camera
- **Sparse-direct visual**: keypoint 없이 photometric matching
- **LiDAR-aided depth**: visual feature에 LiDAR depth 부여
- FAST-LIO 위에 camera 모듈 추가

FAST-LIVO 개요 (Zheng et al., RA-L 2022)

논문
"FAST-LIVO: Fast and Tightly-coupled Sparse-Direct LiDAR-Inertial-Visual Odometry"
IEEE Robotics and Automation Letters, 2022

핵심 기여

- **3-sensor tight coupling**: LiDAR + IMU + Camera
- **Sparse-direct visual**: keypoint 없이 photometric matching
- **LiDAR-aided depth**: visual feature에 LiDAR depth 부여
- FAST-LIO 위에 camera 모듈 추가

목표

- Visual-aided rotation accuracy 향상
- Texture-rich 환경에서 더 안정적
- Dark/texture-less 환경: LiDAR-only fallback

FAST-LIVO 아키텍처

세 센서 순차 통합

1. **IMU Propagation**: ESKF state prediction (5회차 방식)
   - 200Hz IMU → state + covariance 전파
2. **LiDAR IEKF Update**: Point-to-plane (FAST-LIO 방식)
   - ikd-tree + plane fitting + IEKF
   - Geometry 추정 (위치, 자세)
3. **Visual IEKF Update**: Photometric matching
   - LiDAR depth로 visual features 보강
   - Pixel intensity residual
   - Rotation accuracy 향상

FAST-LIVO 아키텍처

세 센서 순차 통합

1. **IMU Propagation**: ESKF state prediction (5회차 방식)
   - 200Hz IMU → state + covariance 전파
2. **LiDAR IEKF Update**: Point-to-plane (FAST-LIO 방식)
   - ikd-tree + plane fitting + IEKF
   - Geometry 추정 (위치, 자세)
3. **Visual IEKF Update**: Photometric matching
   - LiDAR depth로 visual features 보강
   - Pixel intensity residual
   - Rotation accuracy 향상

설계 원칙
LiDAR update **먼저** (geometry 확정)
Visual update **나중** (refinement, 특히 rotation)

Sparse-Direct Visual Front-end

Direct Method 기초

- 특징점(keypoint) 추출/매칭 없음
- 픽셀 intensity 직접 비교 (photometric)
- Sparse: 높은 gradient 영역의 픽셀만 선택

Sparse-Direct Visual Front-end

Direct Method 기초

- 특징점(keypoint) 추출/매칭 없음
- 픽셀 intensity 직접 비교 (photometric)
- Sparse: 높은 gradient 영역의 픽셀만 선택

Residual 정의

Reference frame 패치 $I_{\text{ref}}(p_{\text{ref}})$와
현재 frame에서 예측 위치 패치 $I_{\text{cur}}(p_{\text{cur}})$ 비교:

$$r_c = I_{\text{ref}}(p_{\text{ref}}) - I_{\text{cur}}(p_{\text{cur}}(\hat{T}))$$

Sparse-Direct Visual Front-end

Direct Method 기초

- 특징점(keypoint) 추출/매칭 없음
- 픽셀 intensity 직접 비교 (photometric)
- Sparse: 높은 gradient 영역의 픽셀만 선택

Residual 정의

Reference frame 패치 $I_{\text{ref}}(p_{\text{ref}})$와
현재 frame에서 예측 위치 패치 $I_{\text{cur}}(p_{\text{cur}})$ 비교:

$$r_c = I_{\text{ref}}(p_{\text{ref}}) - I_{\text{cur}}(p_{\text{cur}}(\hat{T}))$$

Jacobian 구조

$$H_{\text{vis}} = \underbrace{\nabla I}_{\text{image gradient}} \cdot \underbrace{\frac{\partial \pi}{\partial p^C}}_{\text{projection}} \cdot \underbrace{\frac{\partial p^C}{\partial \delta x}}_{\text{pose Jacobian}}$$

Visual Feature Patch

Patch 구조

- 각 visual feature: 5×5 또는 7×7 pixel patch
- Reference patch: 최초 관측 시 이미지에서 추출 + 저장
- Current patch: 현재 frame에서 예측 위치에서 추출

Visual Feature Patch

Patch 구조

- 각 visual feature: 5×5 또는 7×7 pixel patch
- Reference patch: 최초 관측 시 이미지에서 추출 + 저장
- Current patch: 현재 frame에서 예측 위치에서 추출

밝기 보정

노출 변화 보정: affine brightness model

$$I_{\text{corrected}} = a \cdot I_{\text{ref}} + b$$

a, b: 밝기 스케일 + offset (관측 시마다 추정)

Visual Feature Patch

Patch 구조

- 각 visual feature: 5×5 또는 7×7 pixel patch
- Reference patch: 최초 관측 시 이미지에서 추출 + 저장
- Current patch: 현재 frame에서 예측 위치에서 추출

밝기 보정

노출 변화 보정: affine brightness model

$$I_{\text{corrected}} = a \cdot I_{\text{ref}} + b$$

a, b: 밝기 스케일 + offset (관측 시마다 추정)

LiDAR-aided Depth

각 visual feature에 LiDAR depth 부여:

$$d = \left({}^{C}_{L}T \cdot p_{\text{LiDAR}}\right)_z$$

Depth 없으면: 무한 depth 가정 (bearing-only)

FAST-LIVO Visual Update 수식

전체 Jacobian

$$H_{\text{vis}} = \nabla I \cdot J_\pi \cdot J_{\text{pose}} \in \mathbb{R}^{1\times 15}$$

각 행렬:

$$\nabla I = \left[\frac{\partial I}{\partial u},\ \frac{\partial I}{\partial v}\right] \in \mathbb{R}^{1\times 2} \quad \text{(image gradient)}$$

$$J_\pi = \frac{\partial \pi}{\partial p^C} = \frac{1}{z}\begin{bmatrix} f_x & 0 & -\frac{u\cdot f_x}{z} \\ 0 & f_y & -\frac{v\cdot f_y}{z} \end{bmatrix} \in \mathbb{R}^{2\times 3}$$

$$J_{\text{pose}} = \frac{\partial p^C}{\partial \delta x} = \left[-\left[{}^C_W R\, p^W\right]_\times,\ {}^C_W R,\ 0,\ 0,\ 0\right] \in \mathbb{R}^{3\times 15}$$

FAST-LIVO IEKF — LiDAR + Visual Combined

순차 Update 전략

1. **LiDAR IEKF** (geometry update):

$$r_L = [\dots],\ H_L \in \mathbb{R}^{N_L \times 15} \Rightarrow P^{(L)},\ \hat{x}^{(L)}$$

2. **Visual IEKF** (refinement):

$$r_V = [\dots],\ H_V \in \mathbb{R}^{N_V \times 15} \Rightarrow P^{(V)},\ \hat{x}^{(V)}$$

FAST-LIVO IEKF — LiDAR + Visual Combined

순차 Update 전략

1. **LiDAR IEKF** (geometry update):

$$r_L = [\dots],\ H_L \in \mathbb{R}^{N_L \times 15} \Rightarrow P^{(L)},\ \hat{x}^{(L)}$$

2. **Visual IEKF** (refinement):

$$r_V = [\dots],\ H_V \in \mathbb{R}^{N_V \times 15} \Rightarrow P^{(V)},\ \hat{x}^{(V)}$$

역할 분담

- LiDAR: 위치/자세 전반
- Visual: rotation accuracy 보완
- IMU: 고주파 motion

Combined 가능

$H = [H_L; H_V]$ 를 한번에
수치 안정성 위해
보통 순차 처리 선호

FAST-LIVO2 (Zheng et al., arXiv 2024; IEEE TRO 2025)

논문
“FAST-LIVO2: Fast, Direct LiDAR-Inertial-Visual Odometry”
IEEE Transactions on Robotics, 2024

FAST-LIVO2 (Zheng et al., arXiv 2024; IEEE TRO 2025)

논문
"FAST-LIVO2: Fast, Direct LiDAR-Inertial-Visual Odometry"
IEEE Transactions on Robotics, 2024

FAST-LIVO 대비 개선점

1. **Visual feature management**: 더 강인한 patch 관리 및 갱신
2. **Online extrinsic calibration**: Camera-LiDAR T_{CL} 온라인 추정
3. **개선된 초기화**: 자동화된 정적/동적 초기화
4. **Multi-camera 지원**: 여러 카메라 동시 활용
5. **더 나은 degeneracy handling**

FAST-LIVO2 (Zheng et al., arXiv 2024; IEEE TRO 2025)

논문
"FAST-LIVO2: Fast, Direct LiDAR-Inertial-Visual Odometry"
IEEE Transactions on Robotics, 2024

FAST-LIVO 대비 개선점

1. **Visual feature management**: 더 강인한 patch 관리 및 갱신
2. **Online extrinsic calibration**: Camera-LiDAR T_{CL} 온라인 추정
3. **개선된 초기화**: 자동화된 정적/동적 초기화
4. **Multi-camera 지원**: 여러 카메라 동시 활용
5. **더 나은 degeneracy handling**

오픈소스
`github.com/hku-mars/FAST-LIVO2`
GPL-2.0, ROS1/ROS2

캘리브레이션 파라미터

- **Spatial**: ${}^{C}_{L}R \in \mathrm{SO}(3)$, ${}^{L}p_C \in \mathbb{R}^3$
- **Temporal**: timestamp offset Δt (LiDAR-Camera 동기)

LiDAR-Camera Extrinsic T_{CL}

캘리브레이션 파라미터

- **Spatial**: ${}^{C}_{L}R \in \mathrm{SO}(3)$, ${}^{L}p_{C} \in \mathbb{R}^3$
- **Temporal**: timestamp offset Δt (LiDAR-Camera 동기)

오프라인 캘리브레이션 도구

- **LCamiCalib** (ETH): 체커보드 + LiDAR
- **Targetless**: 자연 환경에서 자동 추정
- **lidar_camera_calibration** (MIT)

LiDAR-Camera Extrinsic T_{CL}

캘리브레이션 파라미터

- **Spatial**: ${}^{C}_{L}R \in \mathrm{SO}(3)$, ${}^{L}p_{C} \in \mathbb{R}^3$
- **Temporal**: timestamp offset Δt (LiDAR-Camera 동기)

오프라인 캘리브레이션 도구

- **LCamiCalib** (ETH): 체커보드 + LiDAR
- **Targetless**: 자연 환경에서 자동 추정
- **lidar_camera_calibration** (MIT)

FAST-LIVO2 온라인 캘리브레이션

- T_{CL}을 상태벡터에 추가
- IEKF로 함께 추정
- 초기값: 오프라인 결과 사용

Livox MID-360
내장 IMU (BMI088)
별도 IMU 캘리브레이션 불필요
카메라는 별도 장착

Visual-LiDAR Feature Consistency

LiDAR Point $\rightarrow$ Camera

$$p^C = {}^C_L T \cdot p^L = {}^C_L R\, p^L + {}^C t_L$$

$$z = \pi(p^C) = \left[\frac{f_x p^C_x}{p^C_z} + c_x,\ \frac{f_y p^C_y}{p^C_z} + c_y \right]$$

Visual-LiDAR Feature Consistency

LiDAR Point → Camera

$$p^C = {}^C_L T \cdot p^L = {}^C_L R\, p^L + {}^C t_L$$

$$z = \pi(p^C) = \left[\frac{f_x p^C_x}{p^C_z} + c_x,\ \frac{f_y p^C_y}{p^C_z} + c_y \right]$$

Consistency 활용

- LiDAR: 3D 기하구조 (depth, normal)
- Camera: 텍스처/밝기 (gradient)
- 동일 구조를 두 센서로 관측 $\Rightarrow$ 강한 제약

Visual-LiDAR Feature Consistency

LiDAR Point → Camera

$$p^C = {}^C_L T \cdot p^L = {}^C_L R\, p^L + {}^C t_L$$

$$z = \pi(p^C) = \left[\frac{f_x p^C_x}{p^C_z} + c_x,\ \frac{f_y p^C_y}{p^C_z} + c_y \right]$$

Consistency 활용

- LiDAR: 3D 기하구조 (depth, normal)
- Camera: 텍스처/밝기 (gradient)
- 동일 구조를 두 센서로 관측 $\Rightarrow$ 강한 제약

시각적 이점

- Rotation estimation: camera $>$ LiDAR (각도 민감)
- Depth estimation: LiDAR $>$ camera (절대적 정확)
- 융합으로 두 장점 획득

Loop Closure with FAST-LIVO

기본 시스템: No Loop Closure
FAST-LIVO/FAST-LIVO2: 순수 odometry
장거리 이동 시 drift 누적

Loop Closure with FAST-LIVO

기본 시스템: No Loop Closure
FAST-LIVO/FAST-LIVO2: 순수 odometry
장거리 이동 시 drift 누적

Loop Closure 추가 방법

1. **ScanContext + FAST-LIVO**:
   - ScanContext descriptor로 loop detection
   - ICP로 loop constraint 계산
   - Pose graph optimization 별도 수행
2. **FAST-LIVO2 부분 지원**:
   - 제한적 loop closure 기능 내장
3. **LIO-SAM loop closure 모듈 결합**:
   - FAST-LIO2 odometry + LIO-SAM loop detection

Loop Closure with FAST-LIVO

기본 시스템: No Loop Closure

FAST-LIVO/FAST-LIVO2: 순수 odometry
장거리 이동 시 drift 누적

Loop Closure 추가 방법

1. **ScanContext + FAST-LIVO**:
   - ScanContext descriptor로 loop detection
   - ICP로 loop constraint 계산
   - Pose graph optimization 별도 수행
2. **FAST-LIVO2 부분 지원**:
   - 제한적 loop closure 기능 내장
3. **LIO-SAM loop closure 모듈 결합**:
   - FAST-LIO2 odometry + LIO-SAM loop detection

설계 트렌드

LIO backbone + 외부 loop closure module
SLAM = LIO odometry + pose graph correction

FAST-LIVO vs OpenVINS vs LIO-SAM

시스템 비교

항목	FAST-LIVO	OpenVINS	LIO-SAM
센서	L+I+C	C+I	L+I
추정 방법	IEKF	MSCKF EKF	Factor graph
Loop closure	No	No	Yes
Solid-state LiDAR	Yes	–	Partial
Scale 관측	LiDAR 절대	IMU 상대	LiDAR 절대
실시간	Yes (10Hz)	Yes (20Hz)	Yes (10Hz)
CPU 부하	중	낮음	높음
라이선스	GPL	LGPL	BSD

FAST-LIVO vs OpenVINS vs LIO-SAM

시스템 비교

항목	FAST-LIVO	OpenVINS	LIO-SAM
센서	L+I+C	C+I	L+I
추정 방법	IEKF	MSCKF EKF	Factor graph
Loop closure	No	No	Yes
Solid-state LiDAR	Yes	–	Partial
Scale 관측	LiDAR 절대	IMU 상대	LiDAR 절대
실시간	Yes (10Hz)	Yes (20Hz)	Yes (10Hz)
CPU 부하	중	낮음	높음
라이선스	GPL	LGPL	BSD

선택 기준

Loop closure 필요: LIO-SAM
카메라 보조 필요: FAST-LIVO
경량/실시간 우선: FAST-LIO2

Solid-state LiDAR — Livox 특수성

비반복 스캔 패턴

- Livox Avia: 약 70° FOV, rosette 패턴
- Livox MID-360: 약 360° 반구형
- 시간이 지날수록 coverage 밀도 증가 (비반복 누적)
- 기계적 회전 없음 → 진동/고장 감소

Solid-state LiDAR — Livox 특수성

비반복 스캔 패턴

- Livox Avia: 약 70° FOV, rosette 패턴
- Livox MID-360: 약 360° 반구형
- 시간이 지날수록 coverage 밀도 증가 (비반복 누적)
- 기계적 회전 없음 → 진동/고장 감소

장점

- 누적 스캔으로 고밀도
- 기계적 신뢰성
- 낮은 가격
- UAV에 적합 (경량)

FAST-LIO2 지원

- Livox SDK 직접 통합
- 비반복 패턴 처리
- Per-point timestamp
- Undistortion 완전 지원

FAST-LIVO 실전 사용


```
# FAST-LIVO2 설치및실행
cd ~/catkin_ws/src
git clone https://github.com/hku-mars/FAST-LIVO2
cd .. && catkin_make

# 실행 (Livox Avia + camera)
roslaunch fast_livo mapping_avia.launch

# 주요파라미터  (config/avia.yaml)
# lidar_type: 1      # Avia=1, HDL64=2, Ouster=3
# time_sync_en: true # 시간동기화
# filter_size_surf: 0.5  # map voxel size
# cube_len: 1000.0   # local map 크기 [m]
# visual_en: true    # 카메라활성화
# cam_topic: /camera/image_raw
# lidar_topic: /livox/lidar
# imu_topic: /livox/imu
```

권장 하드웨어
IMU: BMI088 (Livox MID-360 내장)

FAST-LIVO2 코드 구조 요약


```
// 메인프로세스루프단순화   ()
void process() {
    // 1. IMU propagation (ESKF, 강5 방식)
    imu_process->Process(measures, kf, feats_undistort);

    // 2. LiDAR IEKF update
    for (int i = 0; i < NUM_ITER; i++) {
        // Point-to-plane residuals 모든( 포인트)
        h_share_model_lidar(kf.get_x(),
                            H_lidar, r_lidar, valid_mask);
        kf.update_iterated_dyn_share_modified(
            r_lidar, H_lidar, R_lidar);
        if (kf.converged()) break;
    }

    // 3. Visual IEKF update 유효한( features)
    for (auto& feat : tracked_visual_features) {
        // Photometric residual
        if (compute_photometric_residual(
                kf.get_x(), feat, H_vis, r_vis)) {
            kf.update_iterated_dyn_share_modified(
                r_vis, H_vis, R_visual);
        }
    }

    // 4. ikd-tree map 업데이트
    ikdtree.Add_Points(feats_down_world->points, true);
}
```

FAST-LIVO 성능 데이터

벤치마크

데이터셋	결과	비고
VIRAL (UAV 실내)	ATE ≈ 0.1 m	실시간
NCL campus (실외)	Robust / 1 km+	다양한 조명
Dark environment	LiDAR-only 전환	카메라 비활성화
Texture-less	LiDAR 우세	카메라 기여 낮음
Fast motion	Visual rotation 보조	IMU+Camera

FAST-LIVO 성능 데이터

벤치마크

데이터셋	결과	비고
VIRAL (UAV 실내)	ATE ≈ 0.1 m	실시간
NCL campus (실외)	Robust / 1 km+	다양한 조명
Dark environment	LiDAR-only 전환	카메라 비활성화
Texture-less	LiDAR 우세	카메라 기여 낮음
Fast motion	Visual rotation 보조	IMU+Camera

센서 기여도

- **위치**: LiDAR $\gg$ Camera
- **Rotation**: Camera $\approx$ LiDAR (텍스처 있을 때)
- **고속 운동**: IMU 주도 (LiDAR 왜곡 방지)

현장 LiDAR 스택 선택 가이드

환경별 추천 시스템

환경/요구사항	추천 시스템
실외 도심, loop closure 필요	LIO-SAM 또는 FAST-LIO2+SC
UAV 실내, 경량	FAST-LIO2 (Livox Avia)
실내 인간 환경 + camera	FAST-LIVO2
대규모 정밀 mapping	HDL-SLAM + GICP
학습/연구	FAST-LIO2 (깔끔한 코드)
저가 LiDAR (16-beam)	FAST-LIO2

현장 LiDAR 스택 선택 가이드

환경별 추천 시스템

환경/요구사항	추천 시스템
실외 도심, loop closure 필요	LIO-SAM 또는 FAST-LIO2+SC
UAV 실내, 경량	FAST-LIO2 (Livox Avia)
실내 인간 환경 + camera	FAST-LIVO2
대규모 정밀 mapping	HDL-SLAM + GICP
학습/연구	FAST-LIO2 (깔끔한 코드)
저가 LiDAR (16-beam)	FAST-LIO2

공통 요구사항

- IMU rate: ≥ 200 Hz
- CPU: Intel i7 이상 (실시간)
- ROS 환경 구성

FAST-LIO2 ROS 토픽

- **Subscribe**: `/livox/imu`, `/livox/lidar`
  또는 `/velodyne_points`, `/imu/data`
- **Publish**: `/cloud_registered`, `/Odometry`, `/path`

FAST-LIO2 ROS 토픽

- **Subscribe**: /livox/imu, /livox/lidar
  또는 /velodyne_points, /imu/data
- **Publish**: /cloud_registered, /Odometry, /path

```
# 주요파라미터  (mid360.yaml)
filter_size_surf: 0.5     # 지도 voxel 크기 [m]
cube_len: 1000.0           # local map cube 크기 [m]
det_range: 450.0           # LiDAR 유효거리  [m]
imu_en: true               # IMU 활성화
acc_cov: 0.1               # 가속도계 noise
gyr_cov: 0.1               # 자이로 noise
b_acc_cov: 0.0001          # 가속도 bias noise
b_gyr_cov: 0.0001          # 자이로 bias noise
```

권장 설정
IMU: ≥ 200 Hz, LiDAR: 10 Hz
IMU-LiDAR 시간 동기: hardware trigger 권장

LiDAR-IMU Calibration

캘리브레이션 파라미터

- **Spatial**: LiDAR-IMU rotation ${}^{I}_{L}R$ + translation ${}^{I}t_{L}$
- **Temporal**: timestamp offset Δt

LiDAR-IMU Calibration

캘리브레이션 파라미터

- **Spatial**: LiDAR-IMU rotation ${}^{I}_{L}R$ + translation ${}^{I}t_{L}$
- **Temporal**: timestamp offset Δt

캘리브레이션 도구

- **lidar_imu_calib** (github.com/ethz-asl):
  핸드헬드 운동 + 자동 최적화
- **FAST-LIVO2 내장**: online extrinsic refinement
- **방법**: 다방향 운동 excitation $\rightarrow$ 최적화

LiDAR-IMU Calibration

캘리브레이션 파라미터

- **Spatial**: LiDAR-IMU rotation ${}^{I}_{L}R$ + translation ${}^{I}t_{L}$
- **Temporal**: timestamp offset Δt

캘리브레이션 도구

- **lidar_imu_calib** (github.com/ethz-asl):
  핸드헬드 운동 + 자동 최적화
- **FAST-LIVO2 내장**: online extrinsic refinement
- **방법**: 다방향 운동 excitation $\rightarrow$ 최적화

Livox 센서
MID-360: 내장 IMU (BMI088)
공장 캘리브레이션 포함
별도 작업 불필요

주의사항
캘리브레이션 오차:
translation >1 cm
rotation $> 0.5°$
$\Rightarrow$ 성능 저하

평가 지표 — LiDAR Odometry

주요 지표

- **ATE** (Absolute Trajectory Error): ATE $= \sqrt{\frac{1}{N}\sum_i \|p_i^{\text{est}} - p_i^{\text{gt}}\|^2}$
- **RPE** (Relative Pose Error): 단거리 상대 오차, segment별 평가
- **APE/RPE**: evo 툴킷 사용

평가 지표 — LiDAR Odometry

주요 지표

- **ATE** (Absolute Trajectory Error): ATE $= \sqrt{\frac{1}{N}\sum_i \|p_i^{\text{est}} - p_i^{\text{gt}}\|^2}$
- **RPE** (Relative Pose Error): 단거리 상대 오차, segment별 평가
- **APE/RPE**: evo 툴킷 사용

evo 사용 예시

```
evo_ape bag result.bag /gt_odom /lidar_odom --align
evo_rpe bag result.bag /gt_odom /lidar_odom -d 1 -u m
```

평가 지표 — LiDAR Odometry

주요 지표

- **ATE** (Absolute Trajectory Error): ATE $= \sqrt{\frac{1}{N}\sum_i \|p_i^{\text{est}} - p_i^{\text{gt}}\|^2}$
- **RPE** (Relative Pose Error): 단거리 상대 오차, segment별 평가
- **APE/RPE**: evo 툴킷 사용

evo 사용 예시

```
evo_ape bag result.bag /gt_odom /lidar_odom --align
evo_rpe bag result.bag /gt_odom /lidar_odom -d 1 -u m
```

주요 데이터셋

- KITTI: 자율주행, 64-beam, GPS ground truth
- MulRan: 도심 반복 주행, 3D Radar 포함
- Hilti SLAM: 건설 환경, mm-수준 ground truth
- NTU VIRAL: UAV 실내, RTK ground truth

SLAM vs Odometry

개념 구분

- **Odometry**: 순차적 pose 추정, 누적 오차 발생
  - FAST-LIO2, OpenVINS
- **SLAM** (Simultaneous Localization and Mapping): Loop closure로 global consistency 유지
  - LIO-SAM, Cartographer

SLAM vs Odometry

개념 구분

- **Odometry**: 순차적 pose 추정, 누적 오차 발생
  - FAST-LIO2, OpenVINS
- **SLAM** (Simultaneous Localization and Mapping): Loop closure로 global consistency 유지
  - LIO-SAM, Cartographer

Odometry 특성

- 낮은 latency
- 실시간 친화적
- Long-term drift
- 단순 구조

SLAM 특성

- Loop closure 교정
- Global consistency
- 높은 계산량
- Factor graph 필요

SLAM vs Odometry

개념 구분

- **Odometry**: 순차적 pose 추정, 누적 오차 발생
  - FAST-LIO2, OpenVINS
- **SLAM** (Simultaneous Localization and Mapping): Loop closure로 global consistency 유지
  - LIO-SAM, Cartographer

Odometry 특성

- 낮은 latency
- 실시간 친화적
- Long-term drift
- 단순 구조

SLAM 특성

- Loop closure 교정
- Global consistency
- 높은 계산량
- Factor graph 필요

트렌드 (2024)
Neural-aided loop detection (PointNetVLAD, BoW3D) + LIO backbone

Neural + LIO 통합 (최신 동향 2024)

LiDAR Neural 연구 방향

1. **NeRF-LiDAR**: Neural implicit map (느림, 높은 정밀도)
2. **3DGS + LIO**: Gaussian Splatting으로 실시간 dense map
3. **Point-LIO**: event-based LiDAR 처리
4. **Wild-Places**: 장기 localization (계절/조명 변화)

Neural + LIO 통합 (최신 동향 2024)

LiDAR Neural 연구 방향

1. **NeRF-LiDAR**: Neural implicit map (느림, 높은 정밀도)
2. **3DGS + LIO**: Gaussian Splatting으로 실시간 dense map
3. **Point-LIO**: event-based LiDAR 처리
4. **Wild-Places**: 장기 localization (계절/조명 변화)

현재 한계
아직 실시간 처리 어려움 (대부분 연구 단계)
실제 로봇: FAST-LIO2 + 전통 방법이 SOTA 실용 스택

Neural + LIO 통합 (최신 동향 2024)

LiDAR Neural 연구 방향

1. **NeRF-LiDAR**: Neural implicit map (느림, 높은 정밀도)
2. **3DGS + LIO**: Gaussian Splatting으로 실시간 dense map
3. **Point-LIO**: event-based LiDAR 처리
4. **Wild-Places**: 장기 localization (계절/조명 변화)

현재 한계

아직 실시간 처리 어려움 (대부분 연구 단계)
실제 로봇: FAST-LIO2 + 전통 방법이 SOTA 실용 스택

주목할 연구

- **KISS-ICP** (Vizzo et al., 2023): 단순하지만 강력한 point-to-point
- **SAGE** (2024): Semantic-aided graph exploration
- **ImMesh** (HKU MARS, 2023): Real-time mesh reconstruction

SLAM 전체 스택 예시 — 자율주행

일반적 자율주행 로컬라이제이션 스택

1. **사전 Mapping**: FAST-LIO2 또는 LIO-SAM으로 HD map 구축
2. **Localization**: ICP로 HD map 대비 실시간 위치 추정
3. **Sensor Fusion**: ESKF (LiDAR + IMU + GPS + Wheel encoder)
4. **Output**: 고정밀 pose (< 10 cm)

SLAM 전체 스택 예시 — 자율주행

일반적 자율주행 로컬라이제이션 스택

1. **사전 Mapping**: FAST-LIO2 또는 LIO-SAM으로 HD map 구축
2. **Localization**: ICP로 HD map 대비 실시간 위치 추정
3. **Sensor Fusion**: ESKF (LiDAR + IMU + GPS + Wheel encoder)
4. **Output**: 고정밀 pose (< 10 cm)

오픈소스 자율주행 스택

- Autoware (ROS2)
- Apollo (Baidu)
- Nav2 (ROS2)

상용 시스템

- Waymo: 독자 스택
- Cruise: 독자 스택
- 공통: LiDAR 중심

Aerial Robot (UAV) LIO 스택

대표 플랫폼

- **센서**: Livox Mid-360 + BMI088 IMU
- **알고리즘**: FAST-LIO2 + FUEL (exploration planner)
- **연구 그룹**: HKUST, ETH Zurich, MIT SPARK

Aerial Robot (UAV) LIO 스택

대표 플랫폼

- **센서**: Livox Mid-360 + BMI088 IMU
- **알고리즘**: FAST-LIO2 + FUEL (exploration planner)
- **연구 그룹**: HKUST, ETH Zurich, MIT SPARK

UAV 특수 도전

- **진동**: IMU noise 증가 $\Rightarrow$ 진동 감쇠 마운트
- **무게**: 전체 센서 패키지 $< 200\,\mathrm{g}$
- **계산**: ARM 기반 컴퓨터 (NVIDIA Orin, Intel NUC)
- **전력**: 배터리 제한 $\Rightarrow$ 효율적 알고리즘 필수

Aerial Robot (UAV) LIO 스택

대표 플랫폼

- **센서**: Livox Mid-360 + BMI088 IMU
- **알고리즘**: FAST-LIO2 + FUEL (exploration planner)
- **연구 그룹**: HKUST, ETH Zurich, MIT SPARK

UAV 특수 도전

- **진동**: IMU noise 증가 $\Rightarrow$ 진동 감쇠 마운트
- **무게**: 전체 센서 패키지 $< 200\,\mathrm{g}$
- **계산**: ARM 기반 컴퓨터 (NVIDIA Orin, Intel NUC)
- **전력**: 배터리 제한 $\Rightarrow$ 효율적 알고리즘 필수

GPS-denied 환경
GPS 없이 실내/실외 자율 비행
FAST-LIO2: 15분 이상 장시간 비행 검증

Ground Robot (AGV/AMR) 스택

환경별 추천

환경	알고리즘	플랫폼 예시
2D 실내	Cartographer, gmapping	TurtleBot
3D 실외	LIO-SAM	Clearpath Husky
3D 실내	FAST-LIO2	Boston Dynamics Spot
대규모 실외	LIO-SAM + SC	Ouster OS0/1

Ground Robot (AGV/AMR) 스택

환경별 추천

환경	알고리즘	플랫폼 예시
2D 실내	Cartographer, gmapping	TurtleBot
3D 실외	LIO-SAM	Clearpath Husky
3D 실내	FAST-LIO2	Boston Dynamics Spot
대규모 실외	LIO-SAM + SC	Ouster OS0/1

속도 우선
FAST-LIO2
저지연, 낮은 CPU
Livox 또는 Ouster

정밀 지도 우선
LIO-SAM
Loop closure 포함
대규모 환경

Ground Robot (AGV/AMR) 스택

환경별 추천

환경	알고리즘	플랫폼 예시
2D 실내	Cartographer, gmapping	TurtleBot
3D 실외	LIO-SAM	Clearpath Husky
3D 실내	FAST-LIO2	Boston Dynamics Spot
대규모 실외	LIO-SAM + SC	Ouster OS0/1

속도 우선
FAST-LIO2
저지연, 낮은 CPU
Livox 또는 Ouster

정밀 지도 우선
LIO-SAM
Loop closure 포함
대규모 환경

AMR 산업 표준
ROS2 Nav2 + AMCL 로컬라이제이션
3D map 기반 localization 증가 추세

회차 요약: LiDAR 상태 추정 스택

5개 핵심 개념

1. **Point-to-plane**: LiDAR 측정 모델의 기본
   - $r_k = n^\top (p_k^W - \bar{q}), H_k = n^\top [-[Rp]_\times, I, 0, \dots]$
2. **IEKF on SO(3)**: 비선형 LiDAR update의 표준
   - 반복 linearization + retraction
3. **ikd-tree**: 동적 map 자료구조
   - $O(\log N)$ insert/delete/query
4. **FAST-LIO2**: 현재 실용 SOTA LIO
   - Feature-free, IEKF, ikd-tree
5. **FAST-LIVO2**: LiDAR+IMU+Camera tight fusion
   - Sparse-direct visual + LiDAR geometry

Modern Robot State Estimation — 전체 지도

강의 전체 구성

1. **1–3회차**: 이론 기초 (Bayes 필터, KF/EKF, SO(3) Lie 군)
2. **4–6회차**: ESKF 완전 구현 (상태 정의, 예측, 업데이트)
3. **7회차**: 최적화 기반 (MAP, Ceres, Factor graph, GTSAM)
4. **8회차**: VIO (MSCKF, FEJ, OpenVINS)
5. **9회차**: LIO (LOAM, LIO-SAM, FAST-LIO2, FAST-LIVO2)

Modern Robot State Estimation — 전체 지도

강의 전체 구성

1. **1–3회차**: 이론 기초 (Bayes 필터, KF/EKF, SO(3) Lie 군)
2. **4–6회차**: ESKF 완전 구현 (상태 정의, 예측, 업데이트)
3. **7회차**: 최적화 기반 (MAP, Ceres, Factor graph, GTSAM)
4. **8회차**: VIO (MSCKF, FEJ, OpenVINS)
5. **9회차**: LIO (LOAM, LIO-SAM, FAST-LIO2, FAST-LIVO2)

핵심 관통 개념

상태 추정 = 노이즈가 있는 센서에서 최적 상태 추론

Bayesian framework → ESKF → IEKF on manifold
모든 방법이 같은 수학적 기반 위에 있음

필터 vs 최적화 — 최종 정리

Filtering 방법 (4–6, 8–9회차)

- ESKF, IEKF, MSCKF
- Sequential Bayesian 업데이트
- $O(n^2)$ per step (state dim)
- 실시간 친화적
- 고정 메모리
- 시스템: FAST-LIO2, OpenVINS

Optimization 방법 (L7)

- MAP, GTSAM, Ceres
- Batch / sliding window
- 더 정확, 더 비용
- Loop closure 자연스럽게 통합
- 시스템: LIO-SAM, ORB-SLAM3

필터 vs 최적화 — 최종 정리

Filtering 방법 (4–6, 8–9회차)

- ESKF, IEKF, MSCKF
- Sequential Bayesian 업데이트
- $O(n^2)$ per step (state dim)
- 실시간 친화적
- 고정 메모리
- 시스템: FAST-LIO2, OpenVINS

Optimization 방법 (L7)

- MAP, GTSAM, Ceres
- Batch / sliding window
- 더 정확, 더 비용
- Loop closure 자연스럽게 통합
- 시스템: LIO-SAM, ORB-SLAM3

실용적 선택
실시간: Filtering (FAST-LIO2, OpenVINS)
최고 정확도: Optimization (GTSAM 기반)
Hybrid: iSAM2 (incremental smoothing, real-time 가능)

주요 오픈소스 정리

핵심 오픈소스 시스템

시스템	방법	센서	언어	라이선스
OpenVINS	MSCKF + ESKF	C+I	C++ ROS	GPLv3
FAST-LIO2	IEKF on SO(3) + ikd-tree	L+I	C++ ROS	GPLv2
FAST-LIVO2	IEKF (LiDAR+Visual+IMU)	L+I+C	C++ ROS	GPLv2
LIO-SAM	IMU preint. + GTSAM iSAM2	L+I	C++ ROS	BSD-3
VINS-Mono	Sliding-window factor graph	C+I	C++ ROS	GPLv3
GTSAM	Factor graph lib	–	C++/Py	BSD
Ceres	NLS solver	–	C++	BSD-3
Cartographer	Pose graph	L	C++ ROS	Apache-2
ORB-SLAM3	Feature SLAM	C+I	C++	GPLv3
Kalibr	Continuous-time batch	C+I	C++/Py	BSD-3

주요 오픈소스 정리

핵심 오픈소스 시스템

시스템	방법	센서	언어	라이선스
OpenVINS	MSCKF + ESKF	C+I	C++ ROS	GPLv3
FAST-LIO2	IEKF on SO(3) + ikd-tree	L+I	C++ ROS	GPLv2
FAST-LIVO2	IEKF (LiDAR+Visual+IMU)	L+I+C	C++ ROS	GPLv2
LIO-SAM	IMU preint. + GTSAM iSAM2	L+I	C++ ROS	BSD-3
VINS-Mono	Sliding-window factor graph	C+I	C++ ROS	GPLv3
GTSAM	Factor graph lib	–	C++/Py	BSD
Ceres	NLS solver	–	C++	BSD-3
Cartographer	Pose graph	L	C++ ROS	Apache-2
ORB-SLAM3	Feature SLAM	C+I	C++	GPLv3
Kalibr	Continuous-time batch	C+I	C++/Py	BSD-3

권장 학습 순서
GTSAM (개념) → FAST-LIO2 (코드) → LIO-SAM (응용)

앞으로의 연구 방향

단기 (1–3년)

- **4D LiDAR**: 포인트별 velocity 측정 ⇒ Doppler LIO
- **Event camera**: 고동적 범위, 낮은 지연
- **Semantic SLAM**: 객체 수준 이해 + 위치 추정

앞으로의 연구 방향

단기 (1–3년)

- **4D LiDAR**: 포인트별 velocity 측정 ⇒ Doppler LIO
- **Event camera**: 고동적 범위, 낮은 지연
- **Semantic SLAM**: 객체 수준 이해 + 위치 추정

중장기 (3–10년)

- **Learning-based**: end-to-end VIO/LIO (DeepLIO)
- **Multi-robot**: collaborative SLAM, 분산 지도 구축
- **Foundation models**: scene understanding + localization 통합

앞으로의 연구 방향

단기 (1–3년)

- **4D LiDAR**: 포인트별 velocity 측정 ⇒ Doppler LIO
- **Event camera**: 고동적 범위, 낮은 지연
- **Semantic SLAM**: 객체 수준 이해 + 위치 추정

중장기 (3–10년)

- **Learning-based**: end-to-end VIO/LIO (DeepLIO)
- **Multi-robot**: collaborative SLAM, 분산 지도 구축
- **Foundation models**: scene understanding + localization 통합

기술적 도전 과제

- Long-term localization (계절, 조명 변화)
- Certification (안전 critical 시스템에서의 신뢰성)
- Compute efficiency (엣지 디바이스에서 실시간)

참고 자료 종합

기초 이론

- Solà et al., “Quaternion kinematics for ESKF” (2017)
- Solà, “A micro Lie theory” arXiv:1812.01537 (2018)
- Forster et al., “On-manifold preintegration” TRO (2017)

참고 자료 종합

기초 이론

- Solà et al., “Quaternion kinematics for ESKF” (2017)
- Solà, “A micro Lie theory” arXiv:1812.01537 (2018)
- Forster et al., “On-manifold preintegration” TRO (2017)

VIO

- Mourikis & Roumeliotis, “MSCKF” ICRA (2007)
- Li & Mourikis, “FEJ-EKF” IJRR (2013)
- Geneva et al., “OpenVINS” ICRA (2020)

참고 자료 종합

기초 이론

- Solà et al., “Quaternion kinematics for ESKF” (2017)
- Solà, “A micro Lie theory” arXiv:1812.01537 (2018)
- Forster et al., “On-manifold preintegration” TRO (2017)

VIO

- Mourikis & Roumeliotis, “MSCKF” ICRA (2007)
- Li & Mourikis, “FEJ-EKF” IJRR (2013)
- Geneva et al., “OpenVINS” ICRA (2020)

LiDAR Odometry

- Zhang & Singh, “LOAM” RSS (2014)
- Shan et al., “LIO-SAM” IROS (2020)
- He et al., “FAST-LIO” RA-L (2021)
- Xu et al., “FAST-LIO2” TRO (2022)
- He et al., “ikd-Tree” RA-L (2022)
- Zheng et al., “FAST-LIVO2” arXiv (2024), IEEE TRO (2025)

참고 자료 종합

기초 이론

- Solà et al., “Quaternion kinematics for ESKF” (2017)
- Solà, “A micro Lie theory” arXiv:1812.01537 (2018)
- Forster et al., “On-manifold preintegration” TRO (2017)

VIO

- Mourikis & Roumeliotis, “MSCKF” ICRA (2007)
- Li & Mourikis, “FEJ-EKF” IJRR (2013)
- Geneva et al., “OpenVINS” ICRA (2020)

LiDAR Odometry

- Zhang & Singh, “LOAM” RSS (2014)
- Shan et al., “LIO-SAM” IROS (2020)
- He et al., “FAST-LIO” RA-L (2021)
- Xu et al., “FAST-LIO2” TRO (2022)
- He et al., “ikd-Tree” RA-L (2022)
- Zheng et al., “FAST-LIVO2” arXiv (2024), IEEE TRO (2025)

필터 ≡ 최적화 등치성

- Bell & Cathey, “IEKF as Gauss-Newton” IEEE TAC (1993)
- Sibley, Sukhatme, Matthies, “Iterated Sigma-point KF” RSS (2006)
- Skoglund, Hendeby, Axehill, “EKF from optimization view” FUSION (2015)

Bayes 정리 한 줄 → 60년의 알고리즘 진화 → 2025년 현장 스택

이 코스에서 얻은 것

- **Part 1** (0–7): ESKF의 모든 줄을 직접 유도하고 OpenVINS `Propagator.cpp`를 읽는 능력
- **Part 2** (7후반–9): MAP 최적화 · MSCKF/VIO · LiDAR — 필터 ≡ Gauss-Newton의 통합 시각
- **현장 스택**: OpenVINS, FAST-LIO/LIVO, GTSAM, VINS-Mono, LIO-SAM 모두 “왜 그렇게 짰는지” 설명 가능

Bayes 정리 한 줄 → 60년의 알고리즘 진화 → 2025년 현장 스택

이 코스에서 얻은 것

- **Part 1** (0–7): ESKF의 모든 줄을 직접 유도하고 OpenVINS `Propagator.cpp`를 읽는 능력
- **Part 2** (7후반–9): MAP 최적화 · MSCKF/VIO · LiDAR — 필터 ≡ Gauss-Newton의 통합 시각
- **현장 스택**: OpenVINS, FAST-LIO/LIVO, GTSAM, VINS-Mono, LIO-SAM 모두 "왜 그렇게 짰는지" 설명 가능

기억해야 할 한 줄

$$\text{ESKF} \subset \text{IEKF} \equiv \text{Gauss-Newton on MAP} \subset \text{Factor Graph}$$

같은 cost를 푸는 서로 다른 스케줄일 뿐.

수고하셨습니다.

APRL · DGIST · Modern Edition