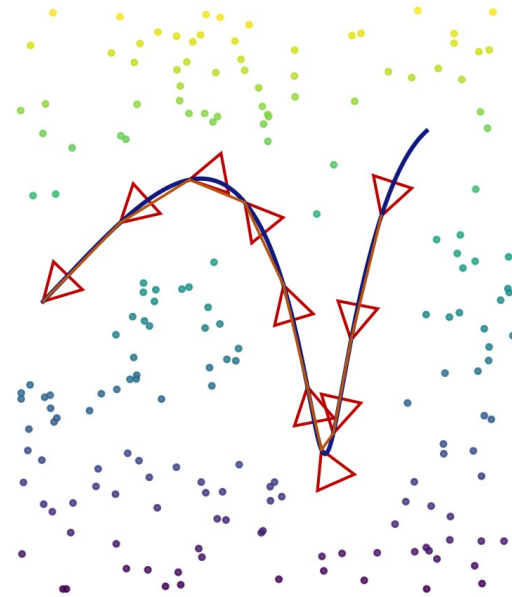


## L9.2 Modern SLAM for AD



Zonghua Gu | Fall 2026

# Roadmap and Learning Outcomes

## Part 1

### Visual SLAM

Features, two-view geometry, PnP, bundle adjustment, ORB-SLAM3

*Lecture A | ~45 min*

## Part 2

### LiDAR and LiDAR-Inertial SLAM

Scan matching, LOAM, LIO-SAM, degeneracy

*Lecture A | ~30 min*

## Part 3

### Localization in AD Stacks

Prior maps, Autoware and Apollo, sensor fusion, error budget

*Lecture B | ~30 min*

## Part 4

### Attacking Localization

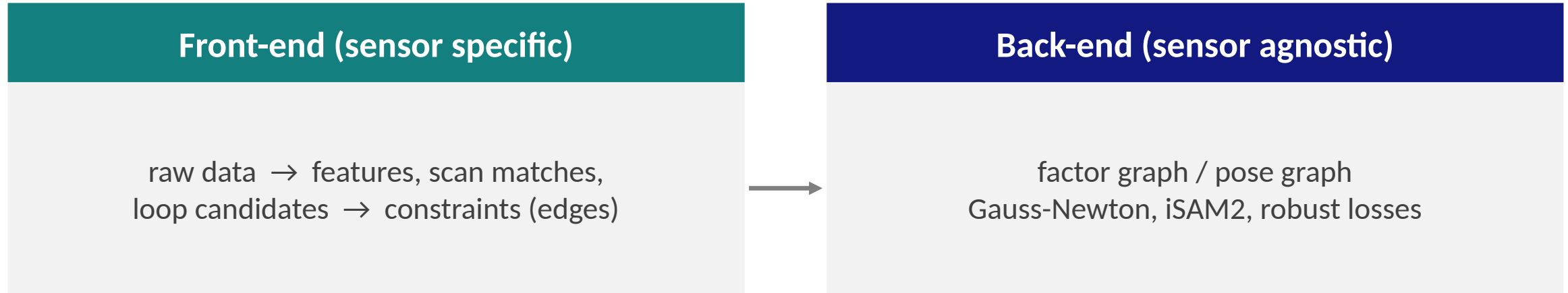
GNSS and LiDAR spoofing, map and loop-closure attacks, defenses

*Lecture B | ~30 min*

## After this lecture you should be able to:

1. Explain how a feature-based visual SLAM system tracks, maps, and corrects drift (PnP, bundle adjustment, loop closure)
2. Describe LiDAR odometry with edge and planar features, and why the IMU and the factor graph make it robust
3. Compare how Autoware and Apollo localize against a prior map and fuse GNSS, IMU, and LiDAR
4. Identify attack surfaces of localization and explain why slow GNSS drift and scan-matching spoofing are hard to detect

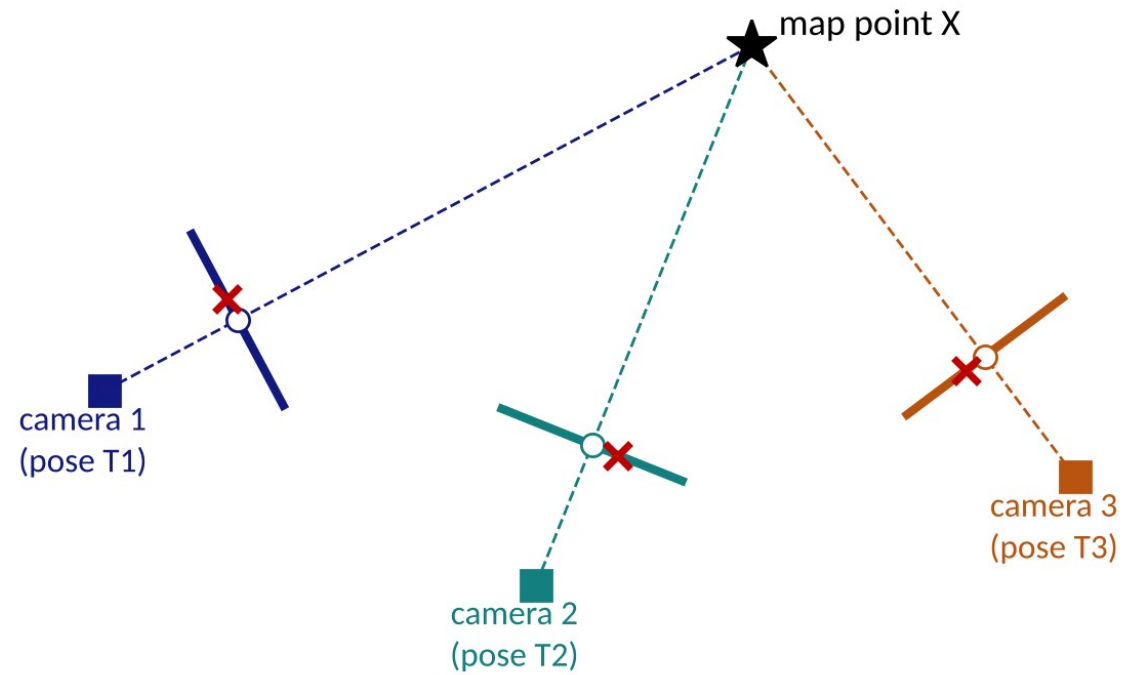
# Recap: Front-End and Back-End



- L9.1 built the back-end: EKF, particles, and **pose graphs** with Gauss-Newton
- Today we look at the **front-ends** that real vehicles use: cameras and LiDAR, alone or with an IMU
- Then we ask how a production AD stack **localizes in a prior map**, and what an attacker can do to it
- Keep one question in mind: which constraint could be **wrong**, and who would notice?

# Part 1: Visual SLAM

- projection  $\pi(T, X)$
- ✗ observed feature  $u$



cameras | features | geometry | bundle adjustment | ORB-SLAM3

# Cameras as SLAM Sensors

- A camera measures **bearing, not range**: a pixel is the projection of a 3-D point along a ray
- Pinhole model: a point  $(X, Y, Z)$  in the camera frame lands on pixel  $(u, v)$
- Cheap, dense, and rich in texture, but depth must be **inferred** from motion, a second camera, or other sensors
- Needs texture and light: weak in dark, blurred, or textureless scenes

$$u = f_x \frac{X}{Z} + c_x, \quad v = f_y \frac{Y}{Z} + c_y$$

$f$  = focal length in pixels,  $(c_x, c_y)$  = principal point; both come from **calibration**.

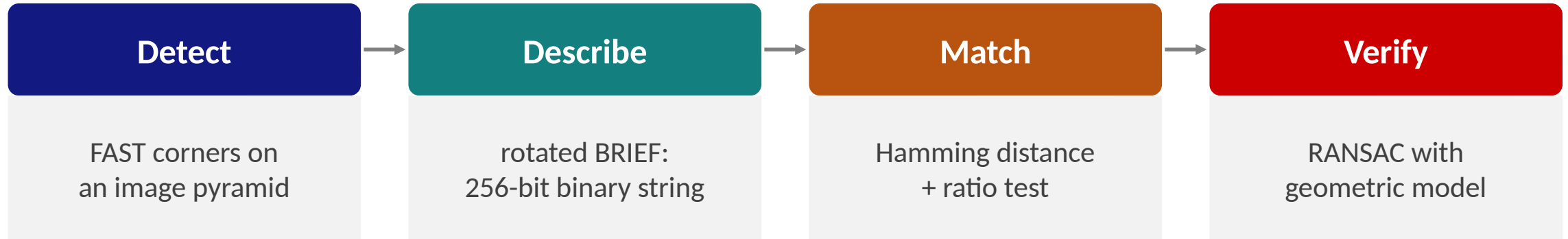
A world point is first moved into the camera frame with the camera pose  $T = (R, t)$ , so each observation depends on the pose **and** the 3-D point: this is the measurement model  $h(x, m)$  of visual SLAM.

# Feature-Based vs. Direct Methods

|                        | Feature-based (indirect)                                            | Direct                                                              |
|------------------------|---------------------------------------------------------------------|---------------------------------------------------------------------|
| <b>Measures</b>        | keypoints and descriptors                                           | pixel intensities                                                   |
| <b>Error minimized</b> | reprojection error (pixels)                                         | photometric error (intensity)                                       |
| <b>Strengths</b>       | robust to lighting change; reliable loop closure and relocalization | uses all gradients; works in weak-texture areas                     |
| <b>Weaknesses</b>      | needs distinct corners; discards most pixels                        | sensitive to exposure and rolling shutter; needs good initial guess |
| <b>Examples</b>        | PTAM, ORB-SLAM family                                               | LSD-SLAM, DSO                                                       |

We follow the **feature-based** pipeline of ORB-SLAM, the most widely used open-source visual SLAM system.

# From Pixels to Matches: ORB



- **ORB** = oriented FAST + rotated BRIEF: scale and rotation tolerant, and fast enough for real time on a CPU
- Binary descriptors are compared with the **Hamming distance** (XOR + popcount): millions of comparisons per second
- **Ratio test**: accept a match only if the best distance is clearly smaller than the second best (rejects repeated texture)
- **RANSAC** removes the remaining outliers by fitting a geometric model (essential matrix or PnP): a few wrong matches can ruin pose estimation

# Two-View Geometry

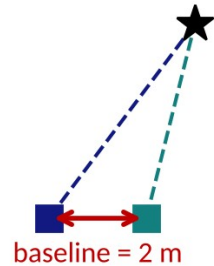
A matched pair of normalized image points  $x_1, x_2$  must satisfy the **epipolar constraint**:

$$x_2^T E x_1 = 0, \quad E = [t]_{\times} R$$

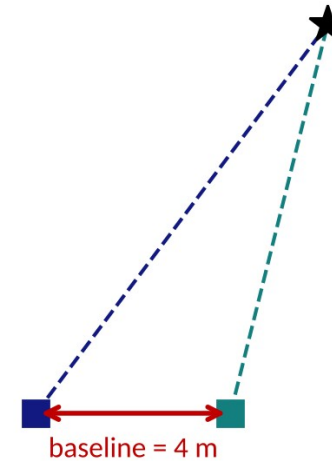
- From  $\geq 5$  matches (5-point algorithm, inside RANSAC) estimate  $E$ , then decompose it into the relative rotation  $R$  and the **direction** of translation  $t$
- **Triangulation**: intersect the two viewing rays to get the 3-D point; accuracy grows with the **baseline** (parallax) between the views
- Failure cases: **pure rotation** (no parallax, no depth) and **planar scenes** ( $E$  is degenerate; a homography  $H$  explains the data)
- ORB-SLAM initializes by computing both  **$H$  and  $F/E$**  in parallel and keeping the better-fitting model

# Monocular Scale Ambiguity

small scene, small motion



scene and motion 2× larger



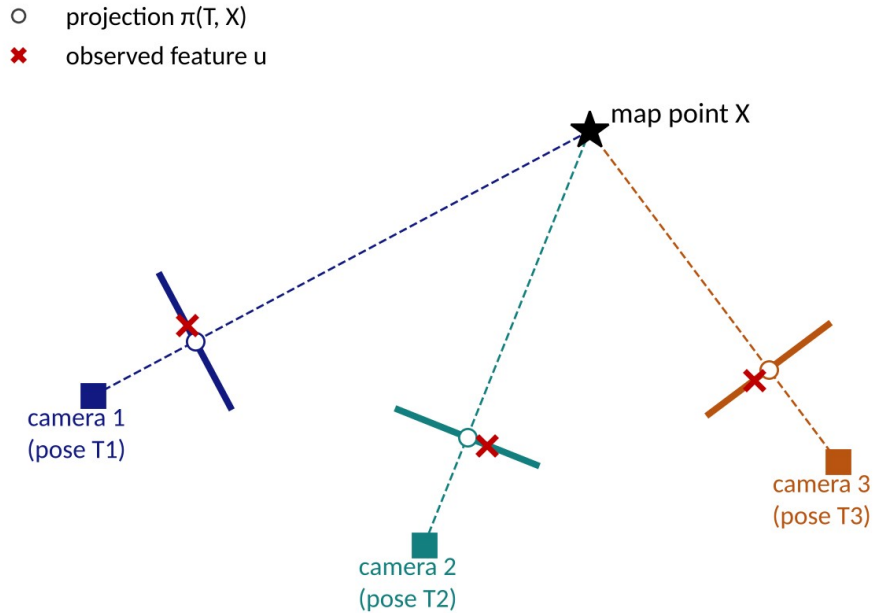
- Scale the whole scene **and** the camera motion by  $s$ : every image stays identical, so one camera cannot recover **metric scale**
- The map and trajectory are correct only **up to a similarity transform** (7 DoF); scale also drifts over time
- Fixes: **stereo** (known baseline), **IMU** (gravity and acceleration), **depth sensor or LiDAR**, wheel odometry, or object sizes
- For AD, metric scale is mandatory: a monocular-only map cannot tell 10 m from 20 m

# Tracking with PnP

---

- Given the previous pose, **project** the local map points into the new frame and match them to features nearby (guided search)
- **PnP** (Perspective-n-Point): from  $n \geq 3$  known 3-D points and their pixels, solve the camera pose; use RANSAC around it
- Then refine the pose with **motion-only bundle adjustment**: points fixed, only the 6-DoF pose is optimized
- Decide whether the frame becomes a **keyframe** (enough new points, enough motion, tracking quality dropping)
- If tracking is lost: **relocalize** with place recognition (bag of visual words) against all keyframes

# Bundle Adjustment



$$\min_{\{T_i\}, \{X_j\}} \sum_{i,j} \rho(\|u_{ij} - \pi(T_i, X_j)\|_{\Sigma}^2)$$

$u$  = observed pixel,  $\pi(T, X)$  = projection of map point  $X$  into the camera with pose  $T$ ,  $\rho$  = robust loss (Huber); the sum runs over all camera–point observations.

This is **full SLAM** from L9.1 with a camera measurement model: nodes are poses **and** points, edges are reprojection errors.

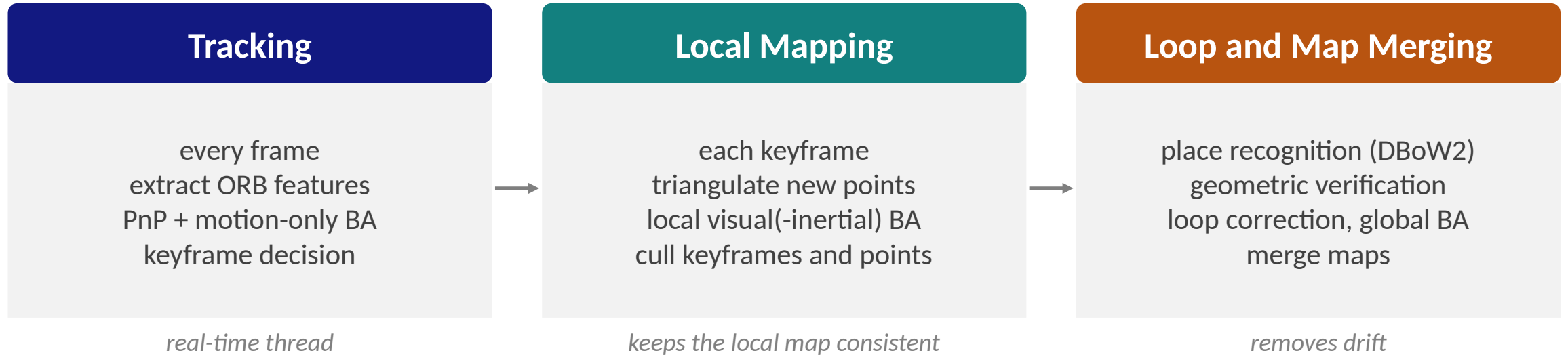
- Solved with Gauss-Newton or Levenberg-Marquardt; the **Schur complement** eliminates the many points first, because each point touches only a few cameras
- **Local BA** (recent keyframes) runs continuously; **global BA** runs after a loop closure

# Keyframes and the Covisibility Graph

---

- Processing every frame in the optimizer is too expensive: keep only **keyframes** that add information
- **Covisibility graph**: keyframes are nodes; an edge links two keyframes that see many of the same map points
- **Local mapping** optimizes only the current keyframe and its covisible neighbors, so the cost stays bounded as the map grows
- **Culling**: redundant keyframes and weak map points are deleted, which keeps the map compact
- The **essential graph** (a sparse spanning subgraph) is used to spread the correction after a loop closure

# ORB-SLAM3 Architecture



**Atlas: a set of disconnected maps; a lost tracker starts a new map and merges it when places are recognized**

- Supports monocular, stereo, and RGB-D cameras, optionally with an **IMU**, with pinhole or fisheye lenses
- Three threads run in parallel, so tracking stays real-time while mapping and loop closing work in the background

# How Accurate Is ORB-SLAM3?

| Benchmark                         | Sensors         | ATE RMSE (m) |
|-----------------------------------|-----------------|--------------|
| EuRoC (drone, 11 sequences, avg.) | monocular       | 0.041*       |
| EuRoC                             | stereo          | 0.084        |
| EuRoC                             | monocular + IMU | 0.043        |
| EuRoC                             | stereo + IMU    | 0.035        |
| TUM-VI room (hand-held, avg.)     | stereo + IMU    | 0.009        |

- \*The monocular average excludes a sequence where tracking failed; trajectories are aligned with a similarity transform first
- Centimeter-level accuracy in **small indoor** scenes with good texture; on large outdoor driving data (e.g., KITTI) drift grows to meters or more
- Reported results are best-case: lighting, motion blur, and dynamic objects in real traffic are much harder

# Adding an IMU: Visual-Inertial SLAM

---

- The IMU gives **metric scale and gravity direction**, and bridges short visual failures (blur, fast rotation)
- The camera corrects the IMU **bias drift**; each sensor covers the other's weakness
- IMU readings between keyframes are **pre-integrated** into one relative-motion factor (no need to re-integrate when poses change)
- ORB-SLAM3 initializes scale with about **5 % error from a 2-second trajectory**, and refines it to about **1 %** after visual-inertial BA at 15 s
- Needs accurate **time sync and extrinsic calibration** between camera and IMU

# Why Visual SLAM Fails on the Road

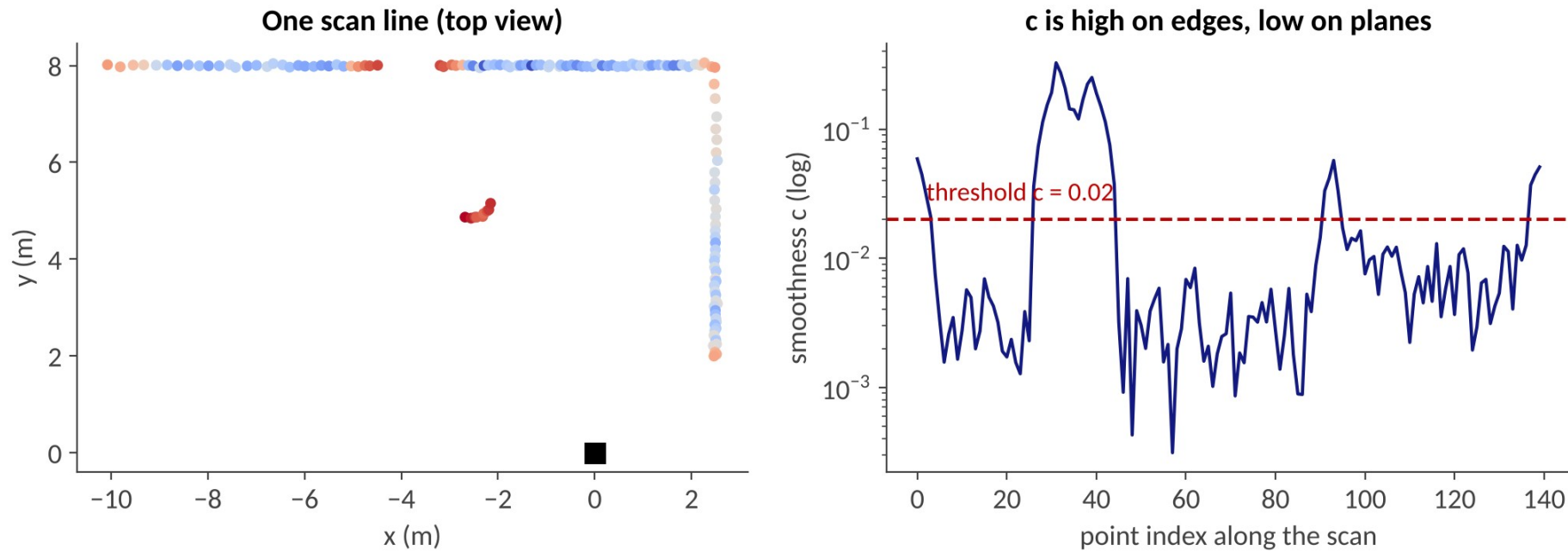
| Condition                          | What breaks                   | Common mitigation                       |
|------------------------------------|-------------------------------|-----------------------------------------|
| Night, glare, tunnel exit          | few or unstable features      | HDR / auto-exposure, IMU, other sensors |
| Fast motion, blur, rolling shutter | matching fails                | global shutter, IMU, higher frame rate  |
| Textureless road or wall           | too few features              | direct methods, LiDAR                   |
| Moving cars and pedestrians        | wrong static-world assumption | semantic masks, RANSAC, robust loss     |
| Repeated structure                 | false loop closure            | geometric verification, robust back-end |
| Straight, forward motion           | low parallax: poor depth      | stereo, LiDAR, wheel odometry           |

# Learning-Based and Neural SLAM

---

- **Learned components** inside a classical pipeline: SuperPoint/SuperGlue features, learned depth and uncertainty
- **DROID-SLAM** (2021): a recurrent network repeatedly updates camera poses and per-pixel depth; a dense bundle-adjustment layer ties them together, with monocular, stereo, and RGB-D variants
- **NeRF / 3-D Gaussian splatting SLAM**: the map is a neural or splat radiance field that is rendered and compared with the images
- Strengths: robustness to blur and low texture, dense maps. Costs: GPU, memory, domain shift, **weak guarantees**
- Today most production AD stacks still rely on **geometric** SLAM and localization with learning used for perception

# Part 2: LiDAR and LiDAR-Inertial SLAM



scan matching | LOAM | LIO-SAM | degeneracy | dynamic objects

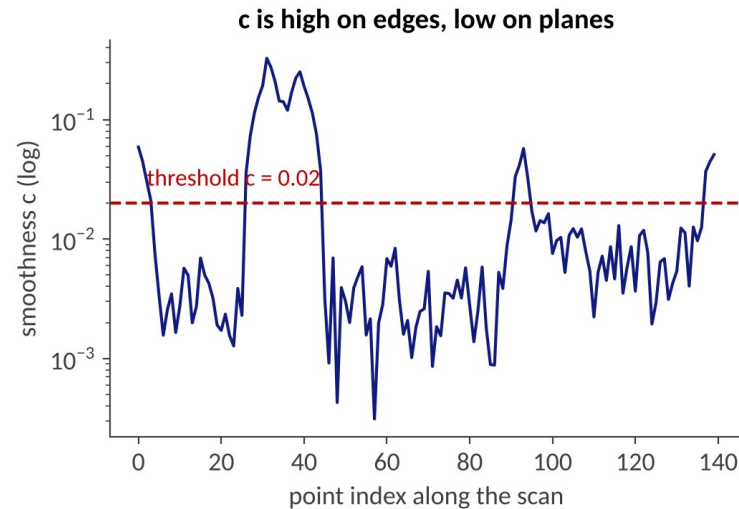
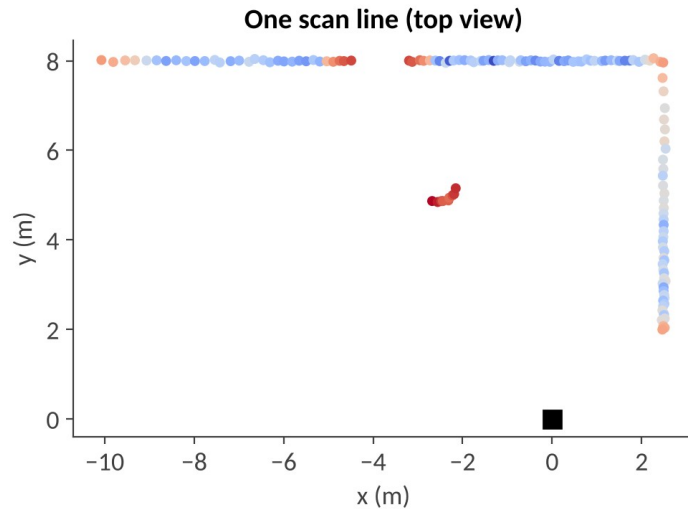
# Scan Registration: ICP and NDT

Find the rigid transform  $(R, t)$  that best aligns the current scan  $P$  to a reference  $Q$  (previous scan or map):

$$\min_{R, t} \sum_i \|n_i^\top (R p_i + t - q_i)\|^2 \quad (\text{point-to-plane ICP})$$

- **ICP**: alternate (1) find nearest neighbors (data association) and (2) solve for the transform; converges only to a **local** minimum, so it needs a good initial guess
- **NDT**: model the reference as a grid of Gaussians and maximize the likelihood of the scan; smooth cost, no point-to-point association; used in Autoware
- The initial guess comes from the **IMU or wheel odometry**; this is why motion prediction and matching are always used together
- The result is a relative-pose **edge** for the pose graph, with a covariance that the optimizer needs (often approximated)
- **Scan-to-scan** matching drifts quickly; **scan-to-map** matching drifts much more slowly (but needs the map in memory)

# LOAM: Edge and Planar Features

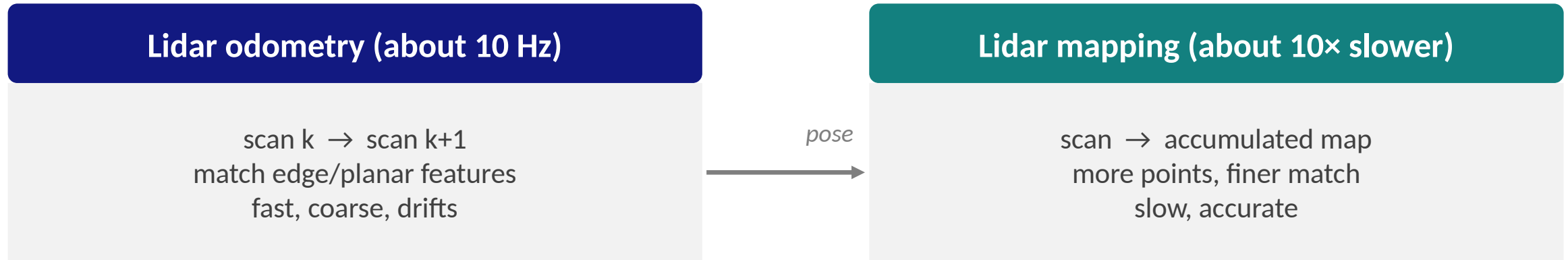


$$c_i = \frac{1}{|S| \|X_i\|} \left\| \sum_{j \in S, j \neq i} (X_i - X_j) \right\|$$

$S$  = the nearest points on the same scan line.  
**Large  $c$**  = edge, **small  $c$**  = plane.

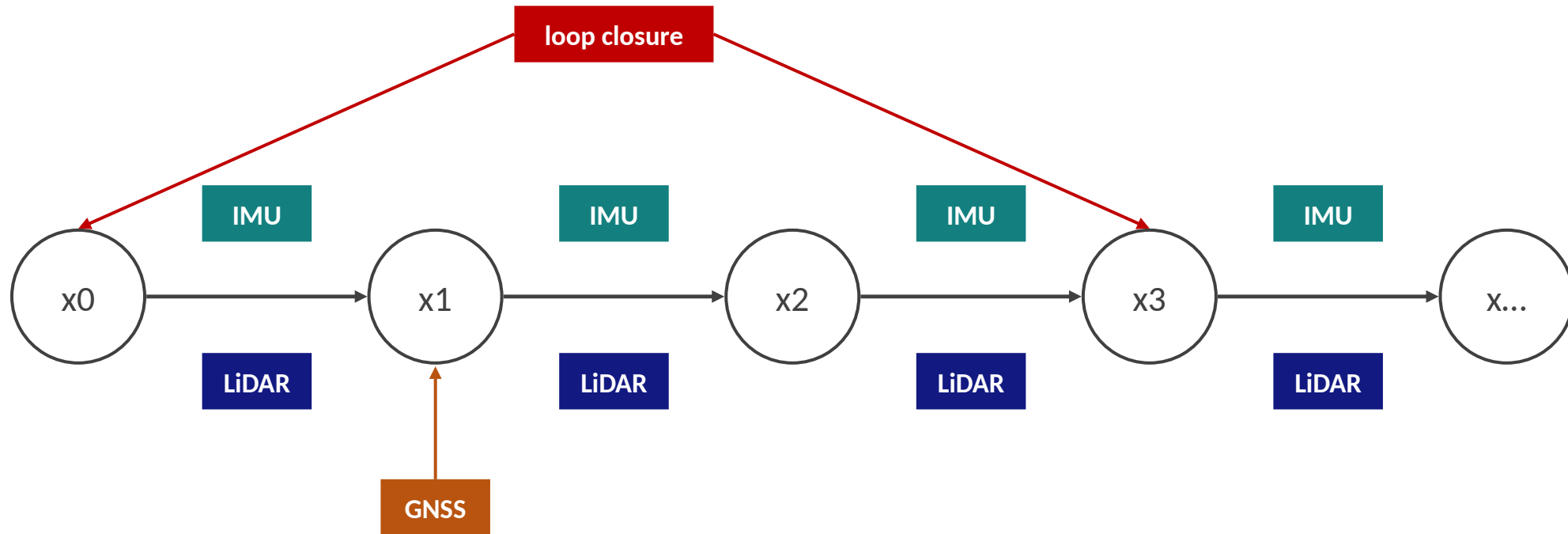
- Match edge points to **lines** and planar points to **planes** in the previous scan or map: the cost is a point-to-line or point-to-plane distance, so most of the cloud is never matched
- In our toy scan with threshold  $c = 0.02$ , about 21 % of points are edge candidates; the rest are planar or discarded
- Features are chosen **per region** of the scan so that matches are spread around the vehicle

# LOAM: Fast Odometry, Slow Mapping



- Splitting the problem into a **high-frequency, low-fidelity** and a **low-frequency, high-fidelity** estimator is what made real-time LiDAR SLAM practical on a CPU
- During one sweep the sensor moves: points are **deskewed** using the motion estimate (otherwise the cloud is distorted)
- LOAM has **no loop closure** and no IMU factor in its original form; later systems (LeGO-LOAM, LIO-SAM, FAST-LIO2) add them
- LOAM ranked first on the KITTI odometry benchmark at the time of publication

# LIO-SAM: The Factor Graph



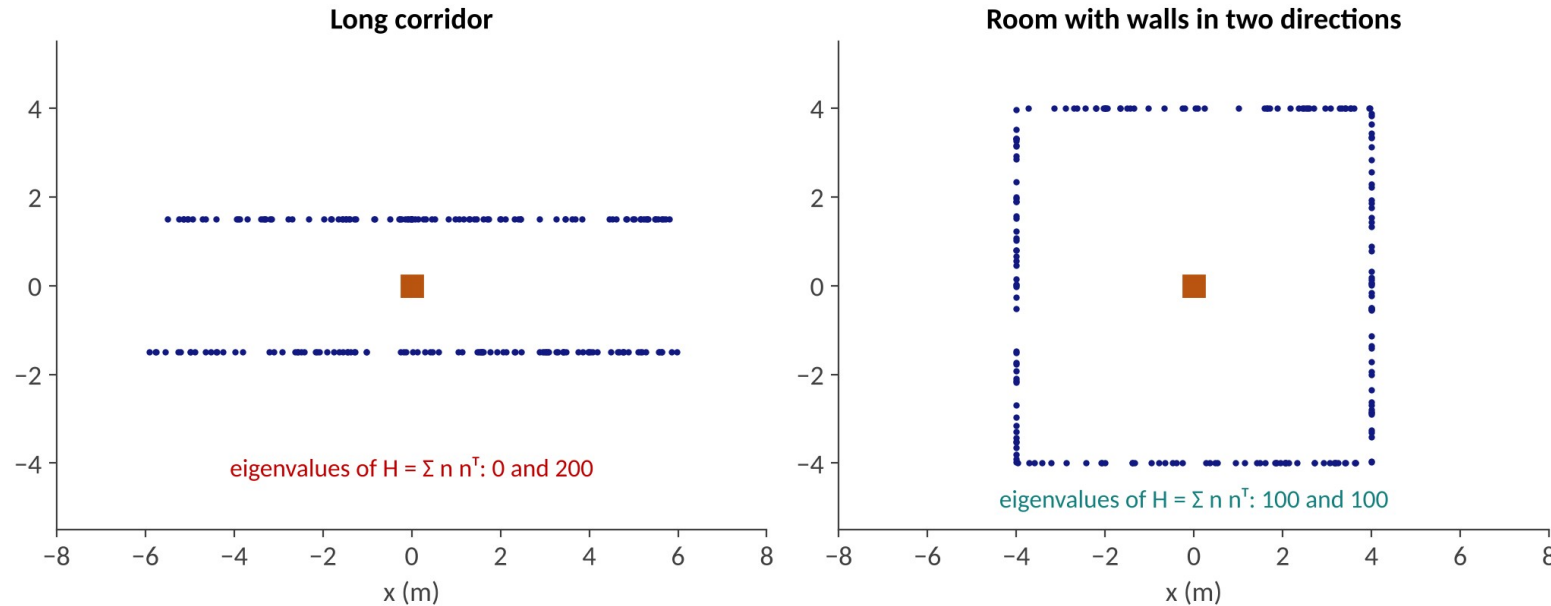
- **Factors:** IMU preintegration, LiDAR odometry (scan-to-local-map), optional GNSS, and loop closure; optimized incrementally with iSAM2
- Scan matching uses a **sliding window of recent keyframes**, not the whole map, which keeps it fast and bounded

# LIO-SAM: What the Factors Buy

| End-to-end translation error (m)           | LOAM   | LIO-odom | LIO-GPS | LIO-SAM |
|--------------------------------------------|--------|----------|---------|---------|
| Campus (1.4 km walk, GPS rarely available) | 192.43 | 9.44     | 6.87    | 0.12    |
| Park (2.9 km, forest trail, UGV)           | 121.74 | 36.36    | 2.93    | 0.04    |

- The metric is the error when the robot **returns to its start**; LIO-odom = IMU + LiDAR factors only, LIO-GPS adds GNSS, LIO-SAM adds loop closure
- Each added factor reduces drift; the **loop closure** gives the largest jump, which is why false loops are so dangerous (L9.1)
- Keyframe rule: new keyframe after **1 m or 10°**; loop candidates within **15 m** of the current state
- Reported for handheld and small-robot data: car-scale speeds, tunnels, and traffic are harder

# Degeneracy in Scan Matching



$$H = \sum_i n_i n_i^T$$

- Each matched plane constrains the translation only along its **normal**;  $H$  collects all constraints
- A **long corridor** has all normals in one direction: eigenvalue 0 along the corridor, so the scan matcher cannot tell how far the car moved
- Tunnels, highways with guardrails, and open fields behave the same way; fix: **IMU**, **wheel odometry**, **GNSS**, or degeneracy-aware solvers

# Dynamic Objects, Maps, and Drift

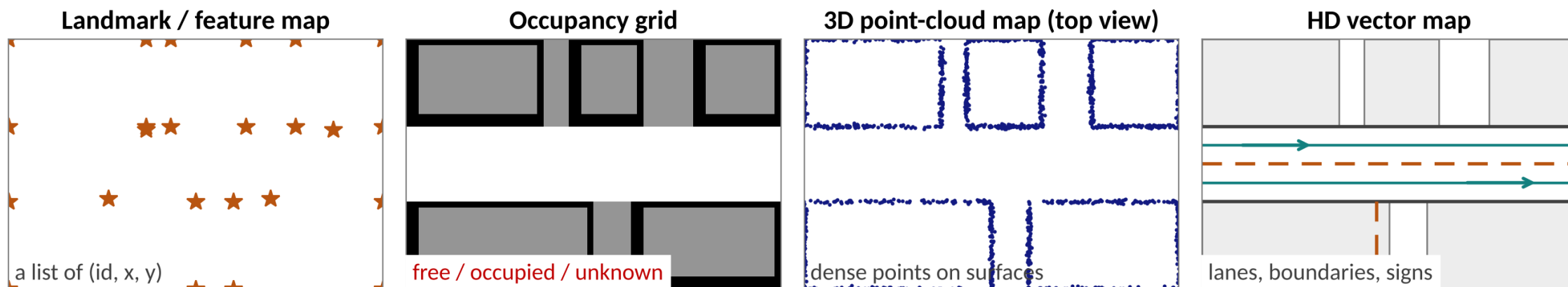
---

- Parked or moving **vehicles and pedestrians** violate the static-world assumption: they create ghost points and wrong matches
- Remedies: **segment and remove** movable classes (semantic or learned), consistency checks across scans, and removal of moving points when building the map
- **Long-term change**: construction, seasons, new buildings; a map older than a few months may no longer match
- Drift per distance is small for LiDAR (typically a **fraction of a percent** of distance) but never zero: loop closure or GNSS is needed
- **Compute**: a 64-beam LiDAR yields  $\sim 10^5$  points per scan; voxel downsampling and feature selection are essential

# Visual vs. LiDAR vs. Fusion

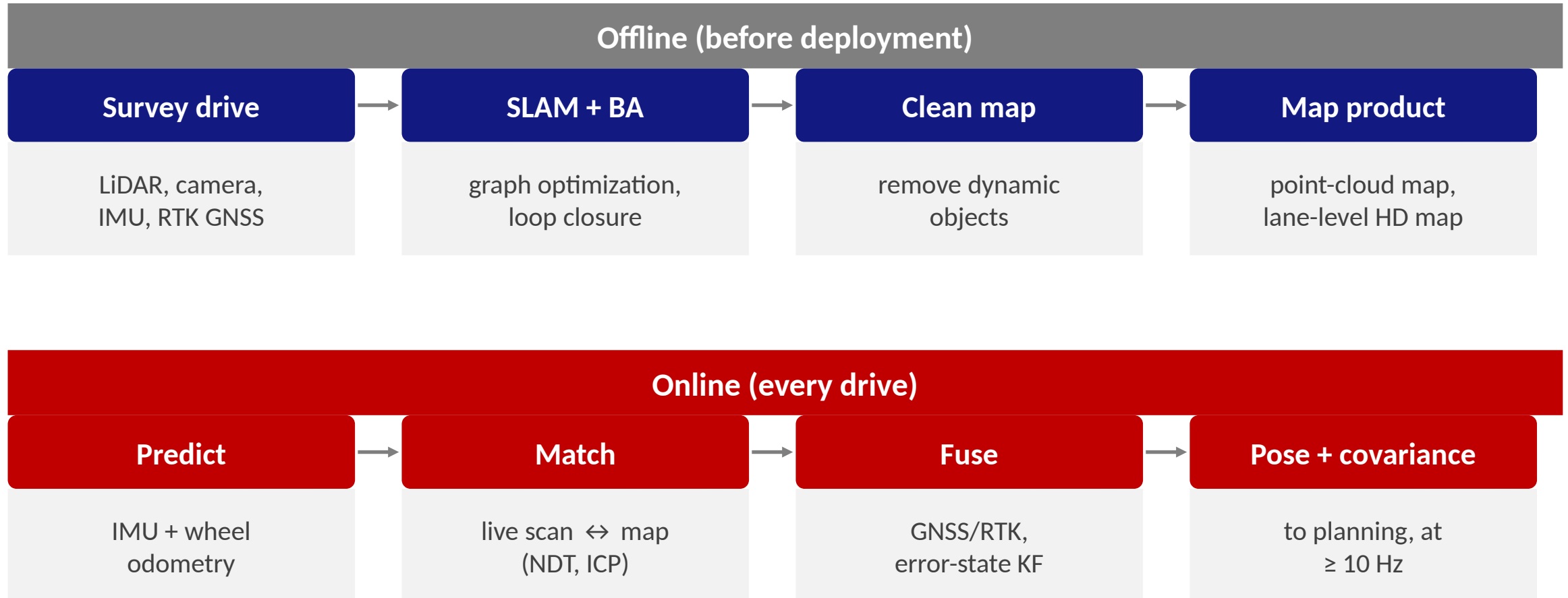
|                           | Camera (visual SLAM)    | LiDAR (LiDAR SLAM)                 | Camera + LiDAR + IMU    |
|---------------------------|-------------------------|------------------------------------|-------------------------|
| <b>Metric scale</b>       | needs stereo or IMU     | direct                             | direct                  |
| <b>Texture / lighting</b> | sensitive               | works at night                     | complementary           |
| <b>Geometry</b>           | sparse, triangulated    | dense, accurate range              | best of both            |
| <b>Weak spots</b>         | dark, blur, textureless | corridors, rain/fog, sparse scenes | calibration, sync, cost |
| <b>Loop closure</b>       | bag of words, cheap     | scan descriptors, more compute     | either, cross-checked   |
| <b>Cost / power</b>       | low                     | high (sensor)                      | highest                 |

# Part 3: Localization in AD Stacks



prior maps | map matching | Autoware and Apollo | fusion | error budget

# From Mapping to Localization



SLAM builds the map **once**, with the best sensors and offline optimization; production vehicles then only **localize** in it.

# Map-Based Localization

---

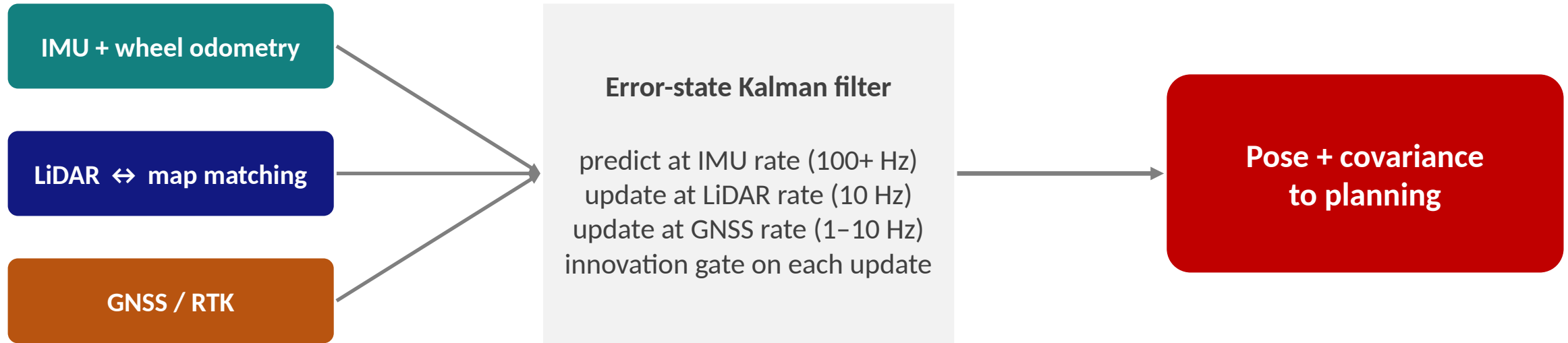
- The prior map turns SLAM into **localization**: drift is no longer integrated, because each scan is anchored to a fixed reference
- **Point-cloud maps**: match the live scan with NDT or ICP; centimeter-level, but maps are large (gigabytes per city)
- **Reflectivity or height maps** (grid of intensity and altitude): smaller, easier to match, used by Baidu Apollo
- **HD vector maps** (lanes, signs, poles): lightweight; localize by matching detected lane markings and landmarks
- Output: pose in the map frame **plus a covariance**, which downstream planning uses to decide how much margin to keep

# Case Study: Autoware vs. Apollo

|                | Autoware (open source)                      | Baidu Apollo (open source)            |
|----------------|---------------------------------------------|---------------------------------------|
| Map used       | 3-D point-cloud map (PCD) + Lanelet2        | LiDAR reflectance/height map + HD map |
| LiDAR matching | NDT scan matcher                            | LiDAR matching against the map grid   |
| Fusion filter  | EKF (pose from NDT, twist from odometry)    | error-state Kalman filter with IMU    |
| GNSS role      | initial pose, optional measurement          | RTK GNSS as a measurement             |
| Outputs        | pose + covariance, with a convergence score | pose + covariance (MSF localization)  |

Both follow the same recipe: **IMU propagation + map matching + GNSS correction in a Kalman filter**. Module names change between releases; check the current documentation.

# Fusion Architecture



- **Error-state** (indirect) filter: estimate the small error of a nominal IMU-integrated state, which keeps the linearization accurate
- The **innovation gate** (Mahalanobis test from L9.1) rejects measurements that disagree with the prediction
- This is also where an attacker aims: whatever passes the gate **moves the pose**

# How Accurate Must Localization Be?

Lateral margin in a lane: a 3.7 m lane and a 1.9 m wide car leave a one-sided clearance of

$$\frac{3.7 - 1.9}{2} = 0.9 \text{ m} \Rightarrow 3\sigma \leq 0.9 \text{ m} \Rightarrow \sigma \leq 0.3 \text{ m}$$

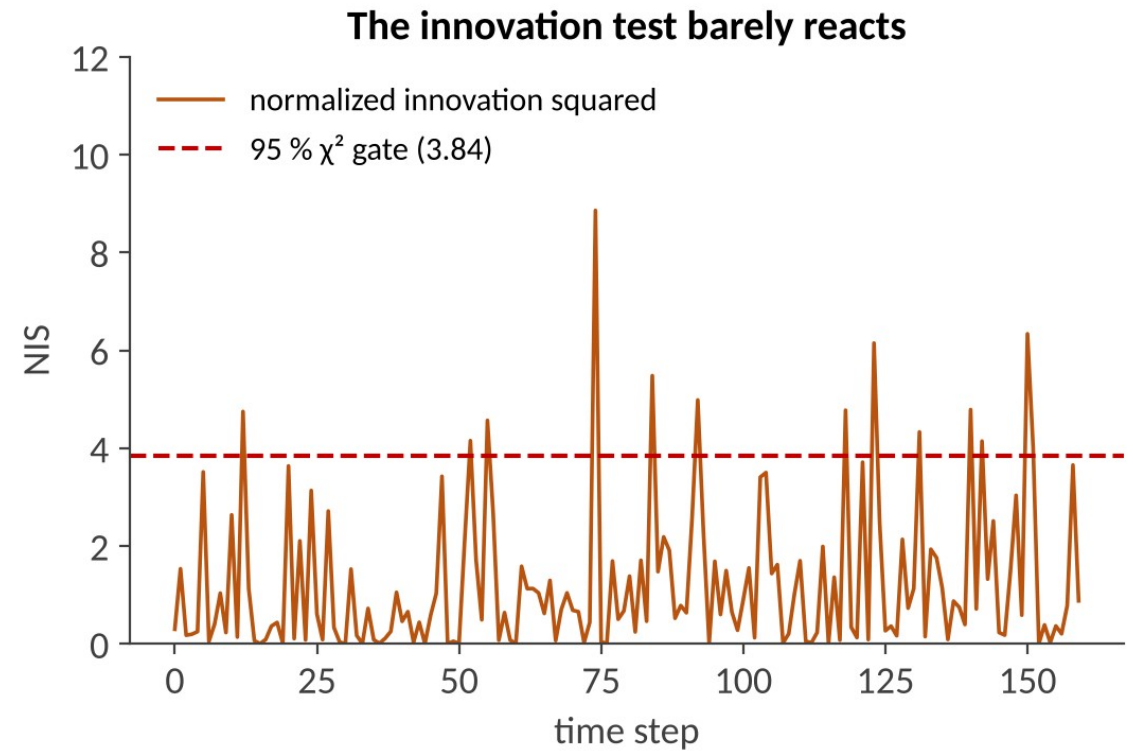
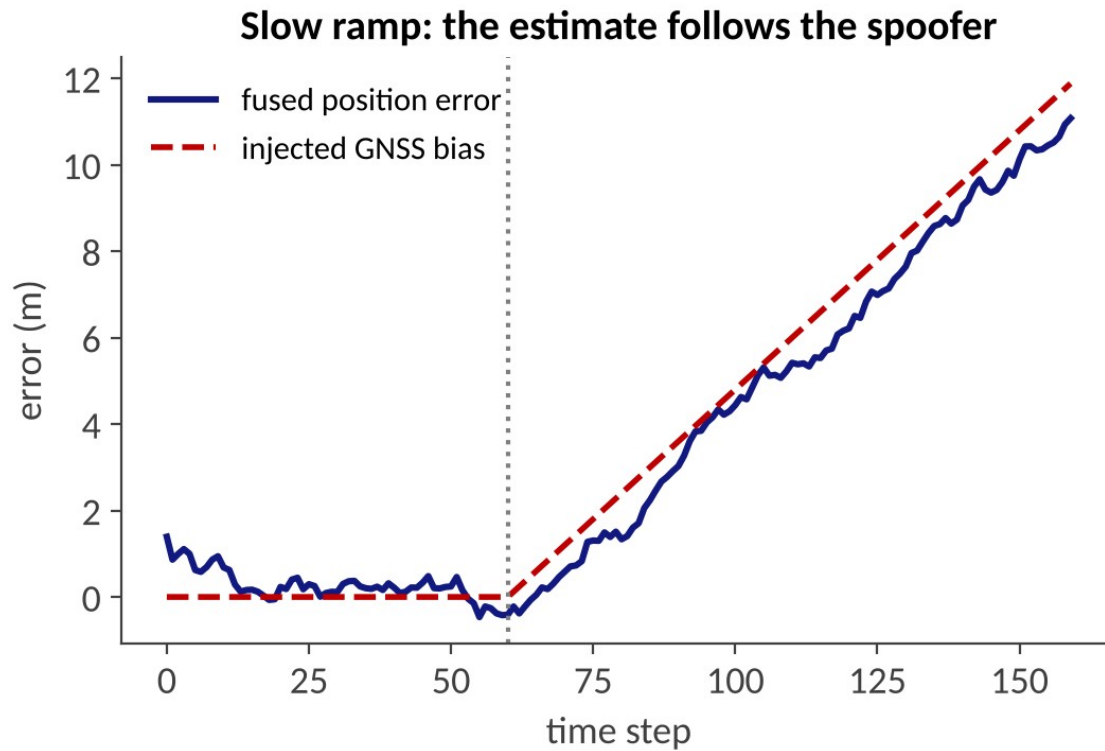
- This is the **absolute** upper bound before the car touches the lane line; planning also needs margin for control error and perception error
- Lane-level driving therefore targets **decimeter-level or better** lateral error; many stacks aim for centimeters
- **Latency** is part of the budget: at 20 m/s a 0.1 s delay equals **2 m** of travel (the same arithmetic as the L4.3 latency budget)
- The **reported covariance** matters as much as the pose: an over-confident filter hides errors from the planner
- Safety cases (ISO 26262 / SOTIF) require that localization failures are **detected** and a safe fallback exists

# When Map-Based Localization Fails

---

- **Map changed:** construction, new buildings, seasonal vegetation; the match score drops or, worse, converges to a **wrong** local optimum
- **Degenerate geometry:** tunnels, long highway walls, open fields; the filter relies on IMU and odometry and drifts
- **GNSS degraded:** multipath in urban canyons produces plausible but wrong positions (not a clean outage)
- **Kidnapped robot:** after a reboot or tunnel, the filter must **re-initialize** globally, not locally
- **Mapless / lightweight maps:** HD maps are costly to build and keep fresh; some stacks use SLAM-style online mapping or navigation maps with a smaller prior
- Discussion: for a 500 m tunnel at 25 m/s, what accumulates, and for how long?

# Part 4: Attacking Localization



attack surface | GNSS and LiDAR spoofing | slow drift | map attacks | defenses

# The Localization Attack Surface

|                               |                                   |                                |
|-------------------------------|-----------------------------------|--------------------------------|
| <b>GNSS</b>                   | <b>LiDAR</b>                      | <b>Camera</b>                  |
| spoofing, jamming             | laser injection, object placement | blinding, adversarial patterns |
| <b>IMU / odometry</b>         | <b>Map</b>                        | <b>Back-end</b>                |
| acoustic or wheel-slip faults | tampering, outdated data          | false loops, gate manipulation |

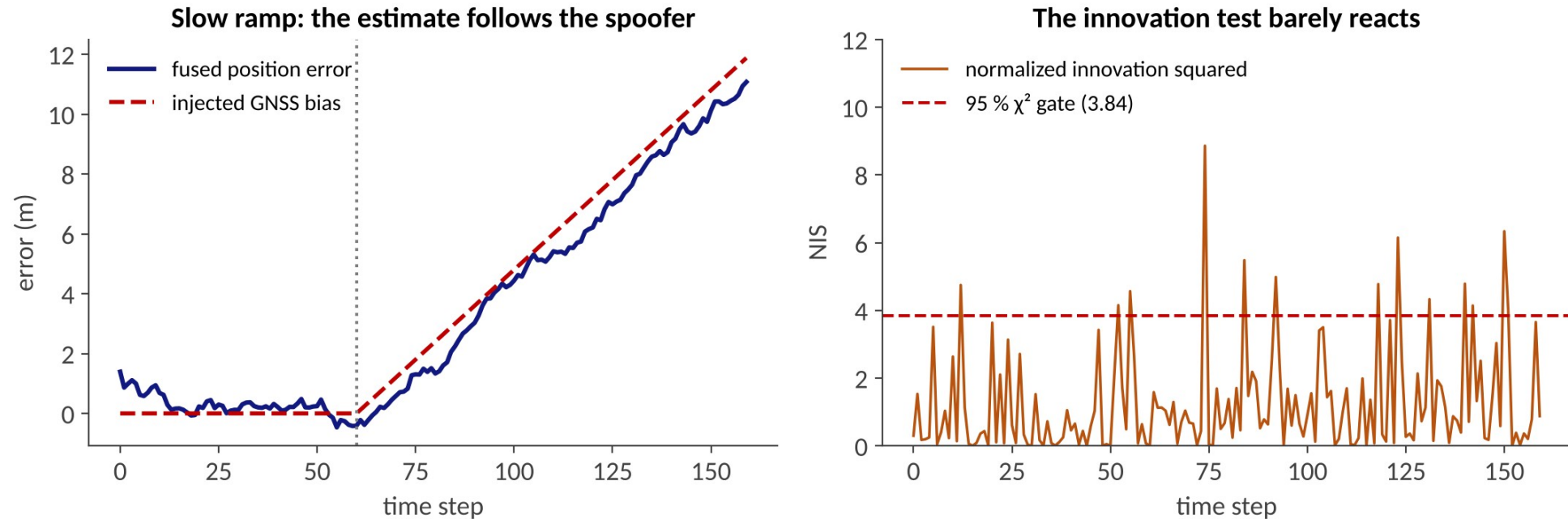
- Attacks can be **physical** (RF, lasers, objects), **data** (map, update channel), or **algorithmic** (exploit the estimator)
- Goal of the attacker: not a crash of the software, but a **small, consistent pose error** that steers the car out of its lane
- A **fusion** system is only as strong as its weakest input that the filter still trusts

# GNSS Spoofing and Sensor Fusion

---

- **Spoofing** transmits fake satellite signals, so the receiver reports an attacker-chosen position; in 2018 Zeng et al. showed **stealthy manipulation of navigation routes** with a portable spoofer
- Multi-sensor fusion was expected to resist it, because IMU and LiDAR disagree with a fake position
- **FusionRipper** (Shen et al., USENIX Security 2020): spoof GNSS in a way that exploits **how a production MSF filter weighs its inputs**, so the fused pose **drifts** away from the truth
- Reported success: at least **97 %** (drive off the road) and **91.3 %** (drive into the wrong way) across six real-world sensor traces
- Message: fusing sensors is **necessary but not sufficient**: the fusion logic itself is an attack target

# Slow Drift vs. the Innovation Gate



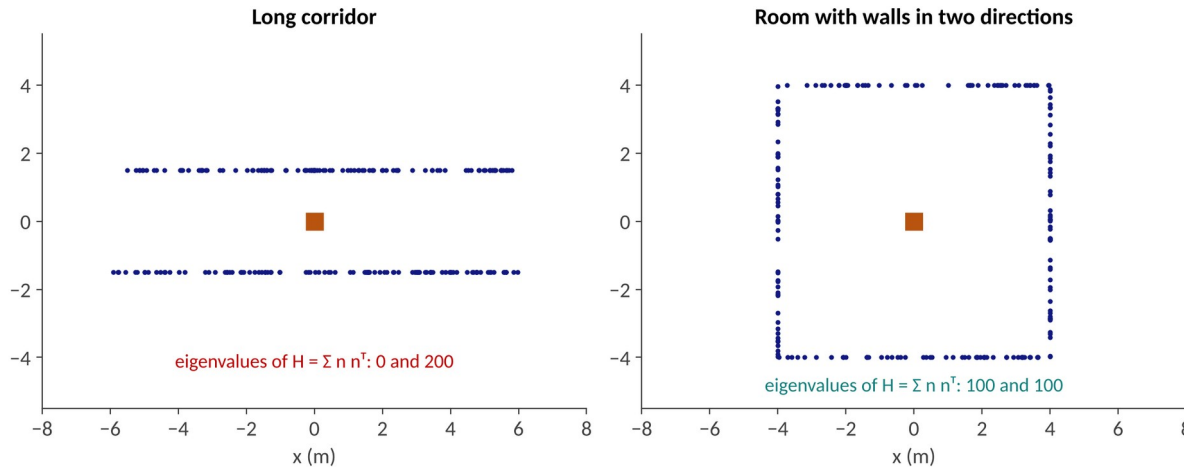
- Toy 1-D filter (IMU + GNSS,  $\sigma = 1$  m): after  $t = 60$  the spoofer adds 0.12 m per step to GNSS; the fused error reaches **11 m** by the end
- Gate exceedance rate: **5 %** before the attack vs. **10 %** during it; the gate sees almost nothing (a 95 % gate flags 5 % by design)
- Reason: each step's disagreement is **within noise**; the filter absorbs it, so the innovation test cannot tell bias from noise. Needs a **drift-aware** check (e.g., compare against LiDAR or odometry over a window)

# LiDAR Spoofing: SLAMSpooF

---

- Attackers can **inject laser pulses** into a LiDAR so that the points the vehicle receives are different from the real scene
- Earlier attacks focused on **perception** (fake obstacles); SLAMSpooF targets **localization** by making scan matching converge to a wrong pose
- The attack uses a **scan-matching vulnerability score** to choose where to place spoofed points that have the most influence
- In real-world tests on ground vehicles it induced position errors of **4.2 m or more** against **three** popular LiDAR-based localization algorithms, more than a lane width
- The authors also discuss countermeasures; this is a young research area

# Why Scan Matching Is Vulnerable



The Hessian  $H = \sum n n^T$  shows **which directions the scan pins down**.

Room: eigenvalues **100 and 100**.

If the two walls facing  $x$  are missing or replaced: **0 and 100**, so  $x$  is no longer constrained.

- Scan matching trusts the **points**: an attacker who controls a subset of points that carry most of the constraint **controls the result**
- Attack-aware reasoning: find the points whose removal or shift changes the optimum the most (high **sensitivity** = high vulnerability)
- The same math explains accidental failures (corridors) and deliberate ones (spoofing): both **shrink the information** in  $H$
- Defense idea: monitor the **eigenvalues and match residuals**, and check consistency with IMU, odometry, and other sensors

# Attacks on Loops, Maps, Timing

---

- **False loop closure:** a spoofed or look-alike place makes the back-end add a wrong edge; in our L9.1 demo one false edge raised ATE from 0.6 m to **12.3 m**
- **Map poisoning:** if map updates, crowd-sourced maps, or OTA channels are not authenticated, the attacker changes the reference everyone localizes against
- **Timing:** delayed or replayed sensor data shifts the pose by  $\text{speed} \times \text{delay}$  ( $20 \text{ m/s} \times 0.1 \text{ s} = \mathbf{2 \text{ m}}$ ); time sync and authenticated timestamps matter
- **Estimator exploits:** manipulate noise covariances, gate thresholds, or sensor trust (e.g., drive the filter to rely on the spoofed source)
- In all cases the pose is **wrong but plausible**: nothing crashes, which is why these attacks are dangerous

# Defenses

| Defense                          | Idea                                                          | Limit                         |
|----------------------------------|---------------------------------------------------------------|-------------------------------|
| Redundancy and cross-checks      | compare GNSS, LiDAR, camera, IMU, odometry                    | fusion logic can be exploited |
| Innovation / residual monitoring | chi-square tests on each update                               | misses slow drift (slide 37)  |
| Windowed drift detection         | compare sensors over seconds, not steps                       | adds delay; threshold tuning  |
| Robust back-end                  | robust loss, switchable constraints, loop verification        | needs good initial guess      |
| Authenticated signals            | GNSS authentication (e.g., Galileo OSNMA), signed map updates | deployment and coverage       |
| Physical-layer hardening         | LiDAR pulse coding, randomized timing, filtering              | cost; attackers adapt         |
| Safe fallback                    | degrade speed, request takeover, stop safely                  | needs a trusted detection     |

# In-Class Activity and Lab Ideas

---

- **Run it** (30 min): run ORB-SLAM3 or a LiDAR SLAM package (e.g., LIO-SAM, FAST-LIO2) on a public sequence (EuRoC, KITTI); compute ATE with evo
- **Compute it** (10 min): lateral error budget and latency distance with your own numbers (see the exercise deck)
- **Simulate an attack** (30 min): extend the 1-D filter on slide 37: change the ramp rate and the gate; find the fastest ramp that is still **undetected**
- **Discussion**: if you were designing the safety monitor for a localization stack, which two signals would you cross-check, and why?

# Summary and Further Resources

## Key takeaways

- **Visual SLAM:** features, PnP, bundle adjustment; monocular scale is unknown without an IMU or stereo
- **LiDAR SLAM:** edge and planar features, dual-rate LOAM; LIO-SAM adds IMU, GNSS, and loop-closure factors
- **Degeneracy** (H has a zero eigenvalue) is the common weakness of scan matching
- AD stacks **localize in a prior map:** IMU + map matching + GNSS in a Kalman filter;  $\sigma \lesssim 0.3$  m
- Localization is a **security-critical** component: small, plausible errors are the real threat

## Further resources

- Campos et al.: **ORB-SLAM3** (T-RO 2021)
- Zhang and Singh: **LOAM**; Shan et al.: **LIO-SAM**
- Barfoot: **State Estimation for Robotics** (free online)
- Shen et al.: **Drift with Devil** (USENIX Security 2020)
- Open source: ORB-SLAM3, LIO-SAM, FAST-LIO2, Autoware, Apollo

# References

---

- [1] C. Campos et al., "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM", IEEE T-RO, 2021. [arxiv.org/abs/2007.11898](https://arxiv.org/abs/2007.11898)
- [2] R. Mur-Artal, J. M. M. Montiel, J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System", IEEE T-RO, 2015.
- [3] E. Rublee et al., "ORB: An efficient alternative to SIFT or SURF", ICCV 2011.
- [4] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge Univ. Press; B. Triggs et al., "Bundle Adjustment: A Modern Synthesis", 1999.
- [5] J. Zhang, S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time", RSS 2014.
- [6] T. Shan, B. Englot, et al., "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping", IROS 2020. [arxiv.org/abs/2007.00258](https://arxiv.org/abs/2007.00258)
- [7] Z. Teed, J. Deng, "DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras", NeurIPS 2021. [arxiv.org/abs/2108.10869](https://arxiv.org/abs/2108.10869)
- [8] G. Wan et al., "Robust and Precise Vehicle Localization based on Multi-sensor Fusion in Diverse City Scenes", ICRA 2018 (Baidu Apollo).
- [9] K. Zeng et al., "All Your GPS Are Belong To Us: Towards Stealthy Manipulation of Road Navigation Systems", USENIX Security 2018.
- [10] J. Shen et al., "Drift with Devil: Security of Multi-Sensor Fusion based Localization in High-Level Autonomous Driving under GPS Spoofing", USENIX Security 2020. [arxiv.org/abs/2006.10318](https://arxiv.org/abs/2006.10318)
- [11] R. Nagata et al., "SLAMSpooF: Practical LiDAR Spoofing Attacks on Localization Systems Guided by Scan Matching Vulnerability Analysis", 2025. [arxiv.org/abs/2502.13641](https://arxiv.org/abs/2502.13641)
- [12] Autoware and Apollo documentation (localization modules). Figures: original simulations and schematics created for this lecture.