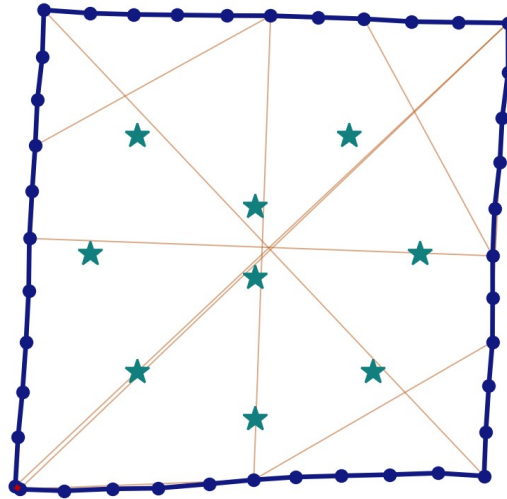


L9.1 SLAM Fundamentals



Zonghua Gu | Fall 2026

Roadmap and Learning Outcomes

Part 1

The SLAM Problem

Why localize, drift, problem definition, models, map types

Lecture A | ~30 min

Part 2

Filtering SLAM

Bayes filter, EKF-SLAM, worked example, data association

Lecture A | ~35 min

Part 3

Particle and Graph SLAM

FastSLAM, pose graphs, Gauss-Newton, sparsity

Lecture B | ~40 min

Part 4

Loop Closure

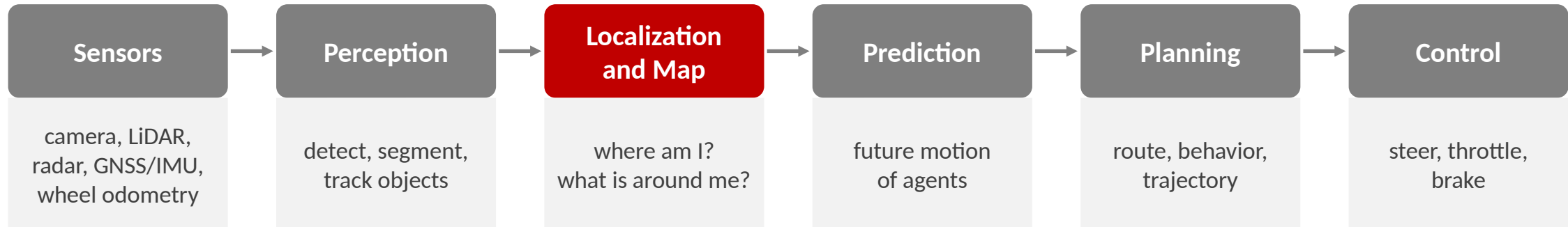
False loops, robust back-ends, ATE and RPE, lab

Lecture B | ~25 min

After this lecture you should be able to:

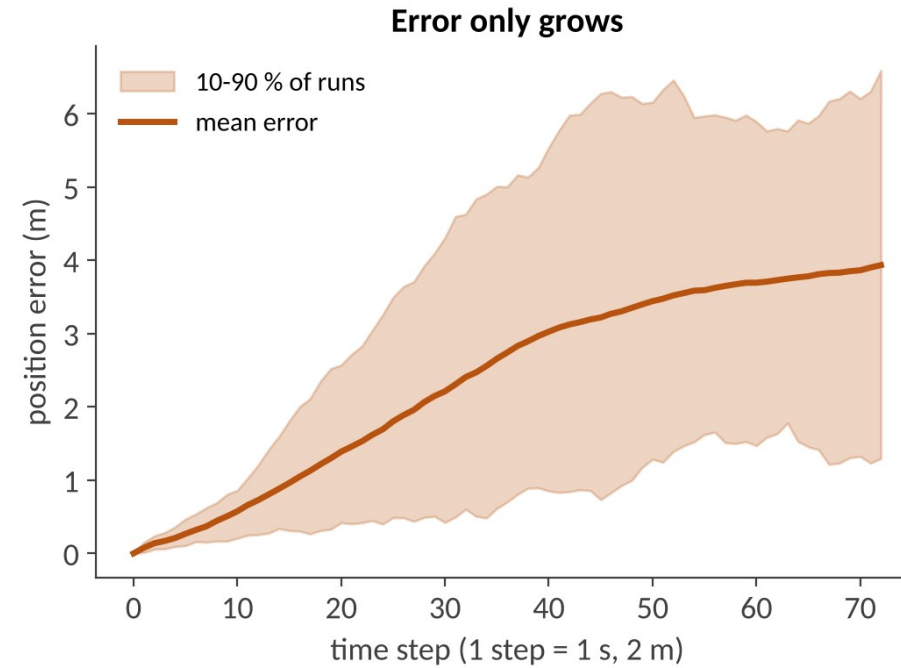
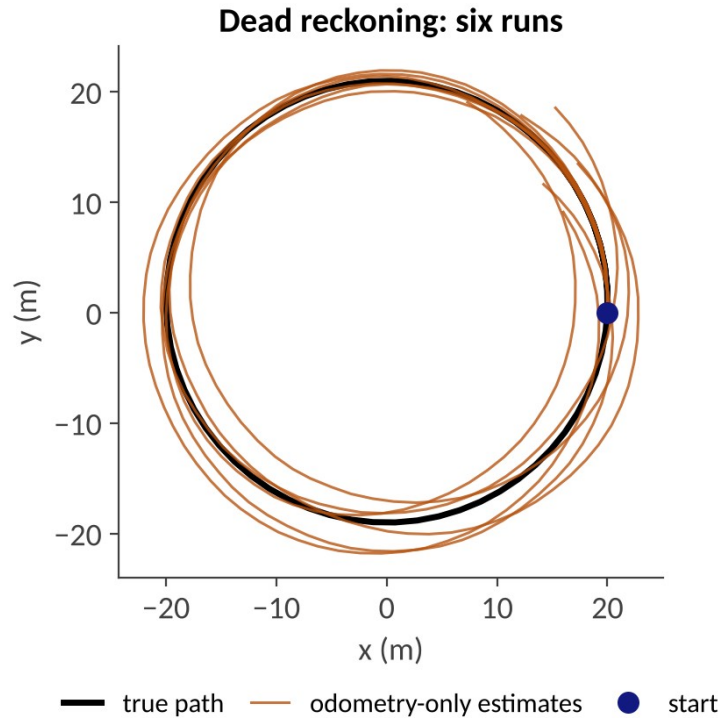
1. Explain why localization and mapping must be solved jointly, and what online and full SLAM estimate
2. Run one EKF-SLAM update by hand and explain why landmark estimates become correlated
3. Formulate a pose graph, build H and b for a small example, and solve it with Gauss-Newton
4. Compare EKF, particle, and graph back-ends, and evaluate a SLAM result with ATE

Where Localization and Mapping Fit



- **Localization** gives every other module the vehicle pose in the map frame; planning and control depend on it
- **Mapping** produces the maps that localization, prediction, and planning rely on: landmark, grid, point-cloud, and HD vector maps
- Today: how a vehicle can build a map **while** localizing in it, with no prior map
- Next lecture (L9.2): modern visual and LiDAR SLAM, map-based localization in AD stacks, and how they can be attacked

Part 1: The SLAM Problem



Why localize | dead reckoning | definitions | models | maps

Why Not Just Use GNSS?

- Standard GNSS errors are several meters; **RTK** can reach centimeters but needs correction data and a clear view of the sky
- Tunnels, parking garages, and urban canyons **block or reflect** the signals (multipath)
- Lane-level driving needs a **decimeter-level lateral error** and a pose expressed in the map frame
- Fusing GNSS with IMU and wheel odometry helps, but dead reckoning still drifts when GNSS drops out
- **Idea:** use what the vehicle sees (landmarks, scans) to correct its own motion estimate, and build the map at the same time

Localization, Mapping, and SLAM

Localization

Given: map m and measurements
Estimate: pose x

e.g., match a LiDAR scan to a prior point-cloud map

Mapping

Given: poses x and measurements
Estimate: map m

e.g., register scans from known poses

SLAM

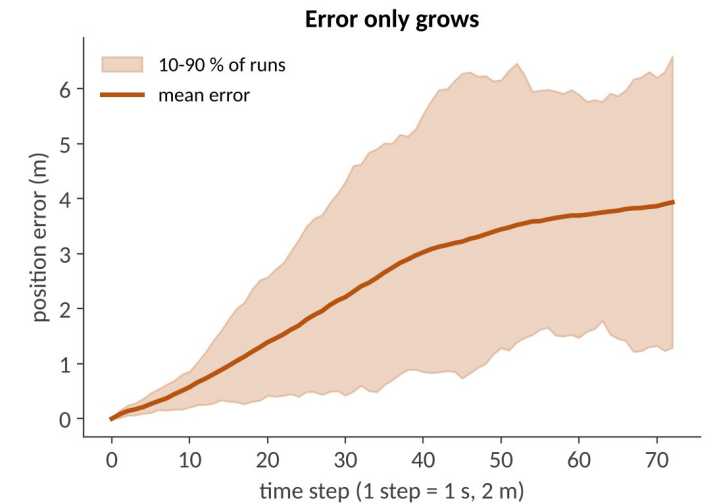
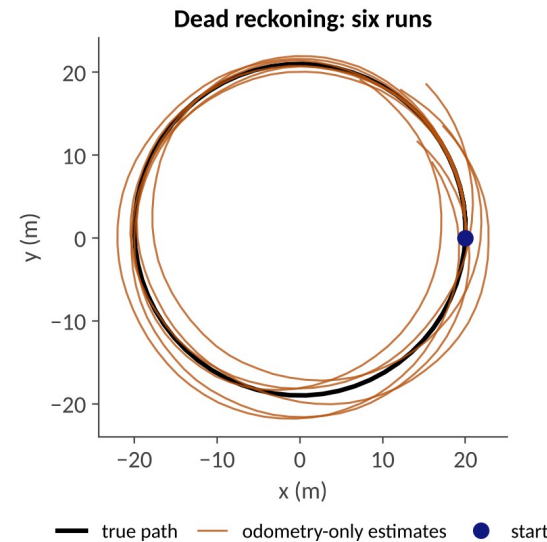
Given: only controls u and measurements z
Estimate: pose x **and** map m

a chicken-and-egg problem

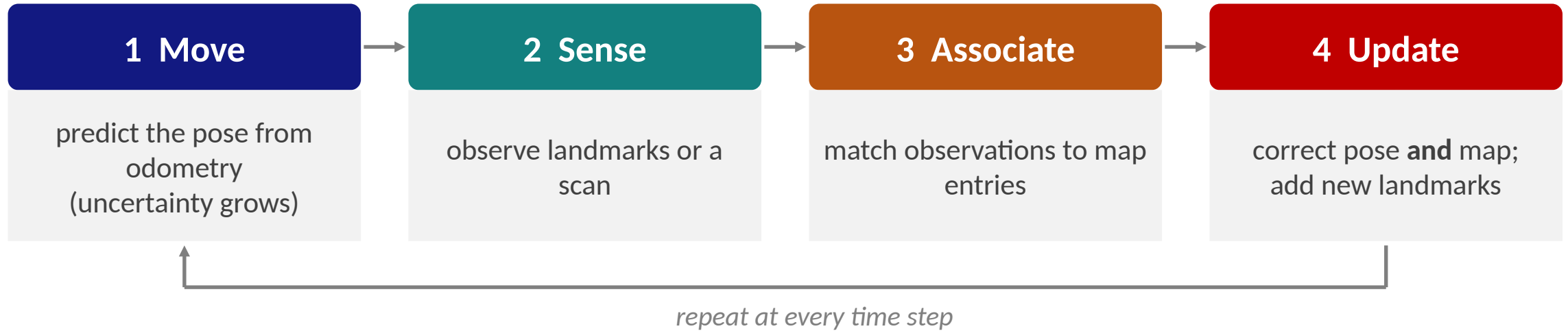
- A good map needs good poses, and a good pose needs a good map
- SLAM solves both **jointly**: every new observation refines the map and the pose
- Typical use cases: first drive in a new area, building HD maps, GPS-denied driving, robots and drones

Dead Reckoning Drifts

- Odometry **integrates** noisy velocity and yaw-rate readings; errors accumulate and never shrink
- In our simulation (circle of radius 20 m, ~1.1 laps, noise 0.1 m/s and 0.02 rad/s) the mean error reaches **3.9 m**
- Heading error is the worst: a small angle error becomes a large position error with distance
- Without an external reference the error is **unbounded**



The SLAM Loop



- Re-observing the **same landmark** from different poses links those poses through the map
- Pose uncertainty and map uncertainty are **coupled**: improving one improves the other
- This loop is the same for EKF-SLAM, FastSLAM, and graph SLAM; only the way the posterior is stored and updated differs

Online vs. Full SLAM

Given the controls u and the measurements z up to time t , estimate the map m and the vehicle pose(s) x

Online SLAM
(filtering)

$$p(x_t, m \mid z_{1:t}, u_{1:t})$$

only the **current** pose; past poses are integrated out
(EKF-SLAM, FastSLAM)

Full SLAM
(smoothing)

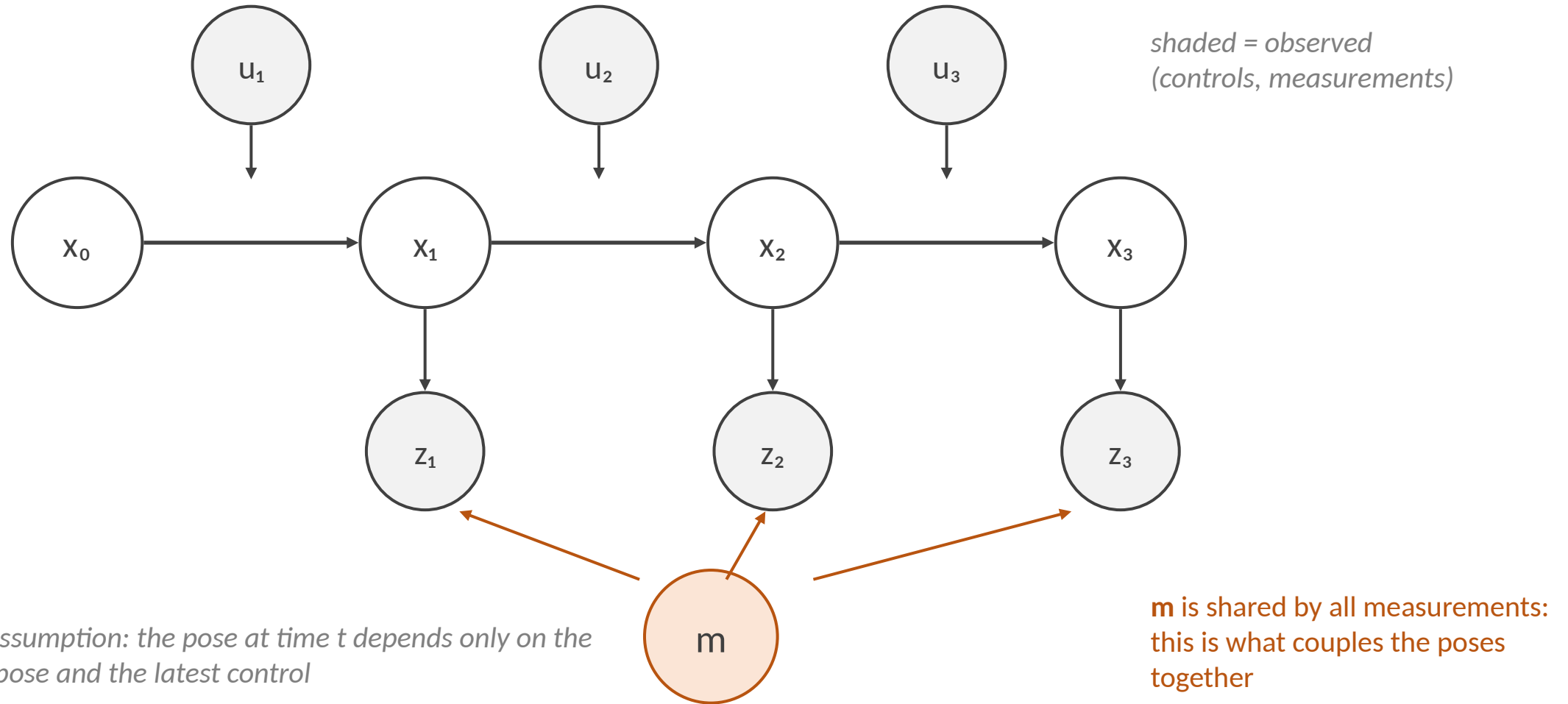
$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$$

the **whole trajectory**
(graph SLAM, bundle adjustment)

$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \int \cdots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) dx_1 \cdots dx_{t-1}$$

- x = poses, m = map, u = controls (odometry), z = measurements; unknown **data association** c says which landmark each z belongs to
- Online SLAM is the full posterior with past poses integrated out; full SLAM keeps them and can revise them later

SLAM as a Graphical Model



Motion and Measurement Models

Motion model from odometry $u = (v, \omega)$, with Gaussian noise of covariance R :

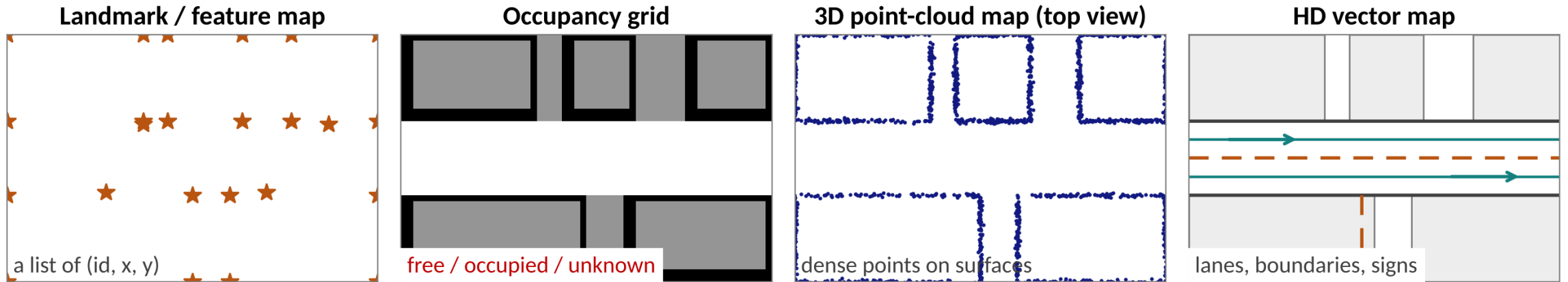
$$x_t = g(x_{t-1}, u_t) + w_t: \quad (x, y, \theta)_t = (x + v\Delta t \cos \theta, \quad y + v\Delta t \sin \theta, \quad \theta + \omega\Delta t)_{t-1}$$

Measurement model: range-bearing to landmark (m_x, m_y) , with noise covariance Q :

$$z_t = h(x_t, m_j) + v_t: \quad r = \sqrt{(m_x - x)^2 + (m_y - y)^2}, \quad \phi = \text{atan2}(m_y - y, m_x - x) - \theta$$

- Both models are **nonlinear**, so the Kalman filter must be linearized (EKF) or replaced by particles or least squares
- Other sensors give other $h(\cdot)$: camera pixel (u, v) , LiDAR scan-to-map overlap, GNSS position
- Noise covariances R and Q come from datasheets or calibration; wrong values make the filter over- or under-confident
- Heading noise in R is the dominant source of drift (see the simulation on slide 7)

What Is a Map?

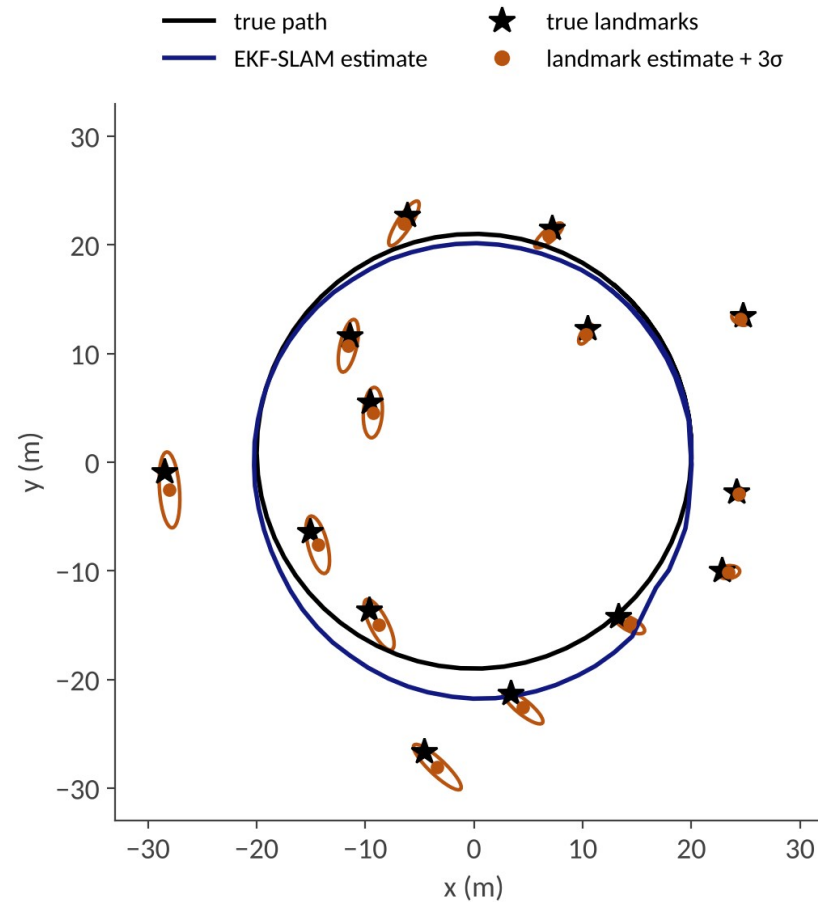


- **Landmark map:** sparse list of (id, x, y); used by EKF-SLAM and FastSLAM; needs a feature extractor
- **Occupancy grid:** free / occupied / unknown per cell; used by planners; memory grows with the area
- **Point-cloud map:** dense 3D points; used for scan-matching localization (ICP, NDT, see L6)
- **HD vector map:** lanes, boundaries, signs; built offline with SLAM plus annotation (see L1.2)

Three Families of SLAM Back-Ends

	Kalman filter (EKF-SLAM)	Particle filter (FastSLAM)	Graph optimization (pose / factor graph)
Posterior	one Gaussian over pose and map	samples of the path; one small Gaussian per landmark	whole trajectory (full SLAM)
Update	online, recursive	online, recursive	batch or incremental least squares
Strength	simple; uncertainty explicit	handles non-Gaussian pose; per-particle data association	accurate; sparse; can revise the past
Weakness	$O(N^2)$ cost; linearization errors	needs many particles; path depletion	needs a good initial guess and loop detection
Today	teaching, small maps, sensor fusion	grid mapping (GMapping)	most modern visual and LiDAR SLAM

Part 2: Filtering SLAM (EKF)



Bayes filter | EKF-SLAM | worked example | data association

Bayes Filter Recap

- Keep a **belief** over the state; update it recursively with two steps
- **Predict**: push the belief through the motion model (uncertainty grows)
- **Correct**: multiply by the measurement likelihood and normalize (uncertainty shrinks)

$$\overline{bel}(x_t) = \int p(x_t \mid x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}$$

$$bel(x_t) = \eta p(z_t \mid x_t) \overline{bel}(x_t)$$

For linear-Gaussian models this is the **Kalman filter**; with Jacobians of g and h it becomes the **EKF**, the same filter used for object tracking in L6. In SLAM the state simply contains the map as well.

EKF Recap

State μ , covariance Σ . Motion g with Jacobian G ; measurement h with Jacobian H ; noise covariances R , Q .

Predict

$$\bar{\mu}_t = g(\mu_{t-1}, u_t)$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$$

Correct

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$$

$$\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$$

- The innovation $v = z - h(\text{predicted mean})$ tells how surprising the measurement is; its covariance is $S = H \Sigma H^T + Q$ (Σ = predicted covariance)
- K balances trust: large Q (noisy sensor) $\rightarrow$ small K ; large predicted Σ (uncertain state) $\rightarrow$ large K

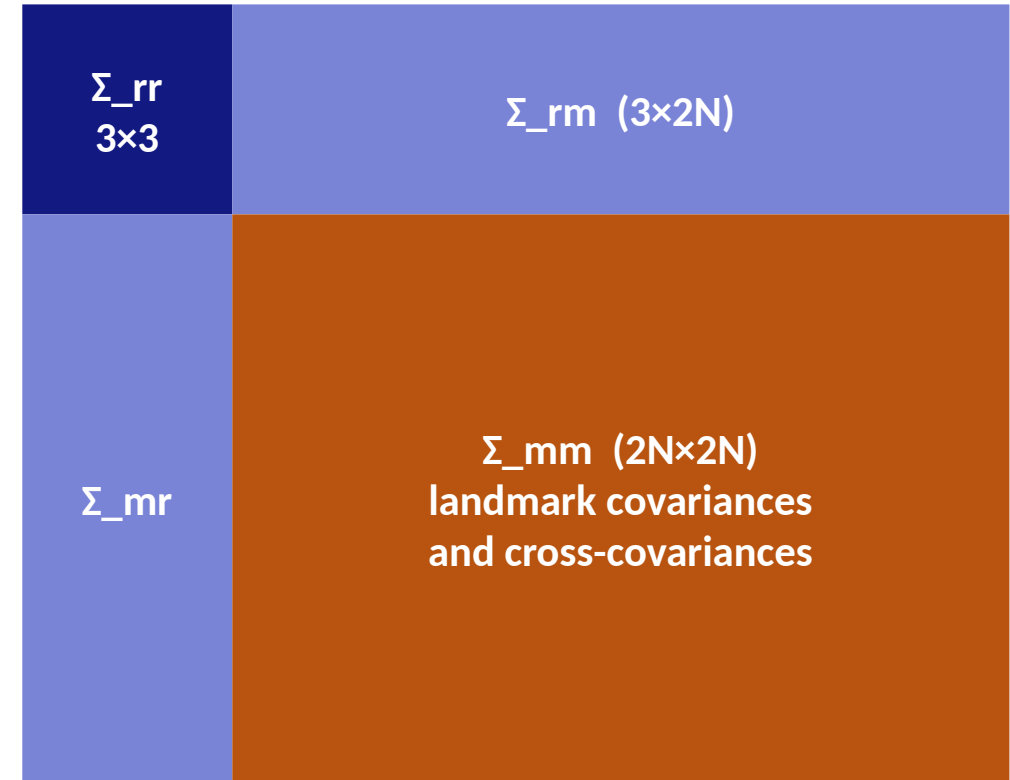
EKF-SLAM: State and Covariance

Stack the robot pose and **all** landmark positions in one Gaussian:

$$\mu = (x, y, \theta, m_{1,x}, m_{1,y}, \dots, m_{N,x}, m_{N,y})^T$$

dimension $3 + 2N$; Σ is a $(3+2N) \times (3+2N)$ matrix

- Σ_{rr} : uncertainty of the robot pose
- Σ_{mm} diagonal blocks: uncertainty of each landmark
- **Off-diagonal blocks**: correlations between pose and landmarks, and between landmarks
- In our simulation with $N = 14$ landmarks: 31×31



$$\Sigma = \begin{bmatrix} \Sigma_{rr} & \Sigma_{rm} \\ \Sigma_{mr} & \Sigma_{mm} \end{bmatrix}$$

EKF-SLAM: Predict Step

Only the robot moves; landmarks are static. The motion Jacobian has a simple structure:

$$\bar{\mu}_r = g(\mu_r, u_t), \quad \bar{\mu}_m = \mu_m$$

$$\bar{\Sigma}_{rr} = G \Sigma_{rr} G^T + R, \quad \bar{\Sigma}_{rm} = G \Sigma_{rm}, \quad \bar{\Sigma}_{mm} = \Sigma_{mm}$$

- G is 3×3 and acts only on the robot block; the landmark block of the full Jacobian is the identity
- **Cost is $O(N)$** : only the robot rows and columns of Σ change
- Σ_{rr} grows by R every step, so the pose uncertainty rises when nothing is observed
- Cross-covariances Σ_{rm} are rotated by G : the map stays correlated with the (moved) robot

EKF-SLAM: Update Step

For each observed landmark j with measurement $z = (r, \varphi)$:

$$\hat{z} = h(\bar{\mu}_r, \bar{\mu}_{m_j}), \quad v = z - \hat{z}, \quad S = H\bar{\Sigma}H^T + Q$$

$$H = [H_r \ 0 \ \cdots \ 0 \ H_{m_j} \ 0 \ \cdots \ 0]$$

- H is **sparse**: nonzero only in the robot columns and in the columns of landmark j
- But the gain $K = \Sigma H^T S^{-1}$ is a full $(3+2N) \times 2$ matrix: **every landmark and the robot are corrected**, even landmarks not observed now
- $\Sigma \leftarrow (I - K H) \Sigma$ touches the entire matrix: **cost $O(N^2)$** per observation
- Angles must be wrapped to $(-\pi, \pi]$ when forming v

Adding a New Landmark

First sighting of a landmark: initialize it from the pose and the measurement (inverse observation model).

$$m_x = x + r \cos(\theta + \varphi), \quad m_y = y + r \sin(\theta + \varphi)$$

- Append two rows and columns to μ and Σ ; new covariance block = $J_x \Sigma_{rr} J_x^T + J_z Q J_z^T$
- The new landmark is **correlated with the robot** from the start (through $J_x \Sigma_{r,\cdot}$), so it inherits the pose uncertainty
- Its covariance is therefore large when the robot pose is uncertain, even if the sensor is precise
- Practical systems add a landmark only after it is seen several times, to avoid initializing outliers

Worked Example: 1-D EKF-SLAM

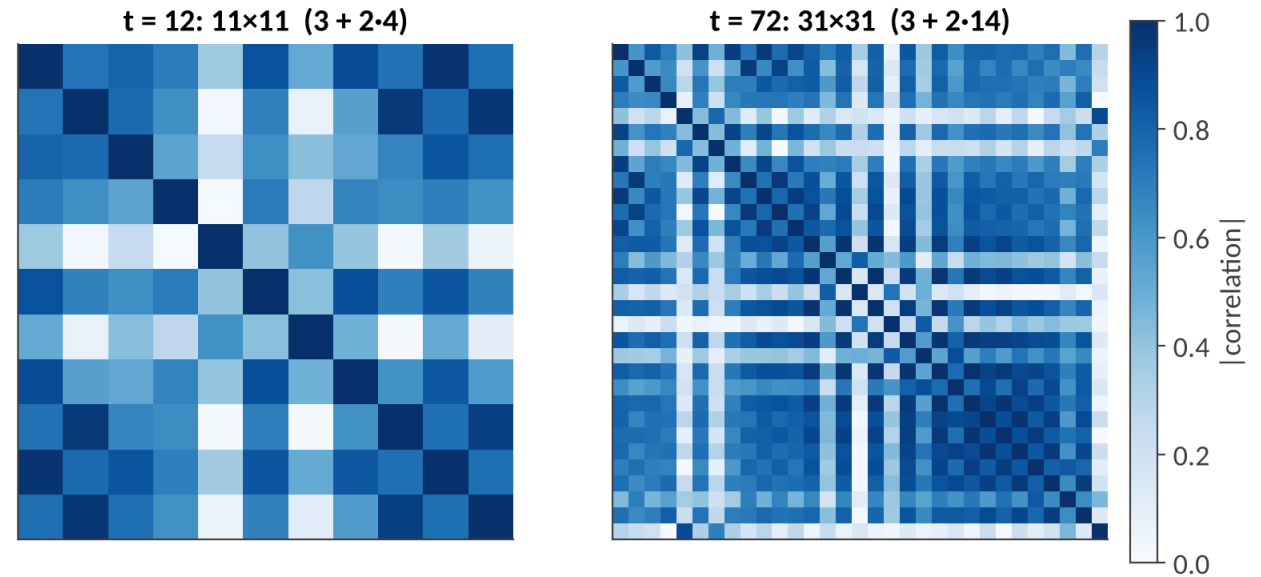
Robot on a line, state $(\mathbf{x}, \mathbf{m})$. Start $x = 0$ (known), first range gives $m = 5.1$ with variance 0.04 . Then move $u = +1$ ($Q = 0.04$) and measure $z = m - x = 3.9$ ($R = 0.04$).

Step	Mean $(\mathbf{x}, \mathbf{m})$	Covariance Σ
0 initialize landmark	$(0, 5.1)$	$[[0, 0], [0, 0.04]]$
1 predict (move +1)	$(1, 5.1)$	$[[0.04, 0], [0, 0.04]]$
2 innovation $v = 3.9 - (5.1 - 1) = -0.2$	$S = 0.04 + 0.04 + 0.04 = 0.12$	$K = (-0.333, +0.333)$
3 update	$(1.067, 5.033)$	$[[0.0267, 0.0133], [0.0133, 0.0267]]$

- The robot moved $+1.067$ (not $+1$): the measurement says the landmark is closer, and the filter splits the blame between $\mathbf{x}$ and $\mathbf{m}$
- Both variances dropped from 0.04 to 0.0267 , and a **correlation** $\rho = 0.0133 / 0.0267 = \mathbf{0.5}$ appeared between $\mathbf{x}$ and $\mathbf{m}$

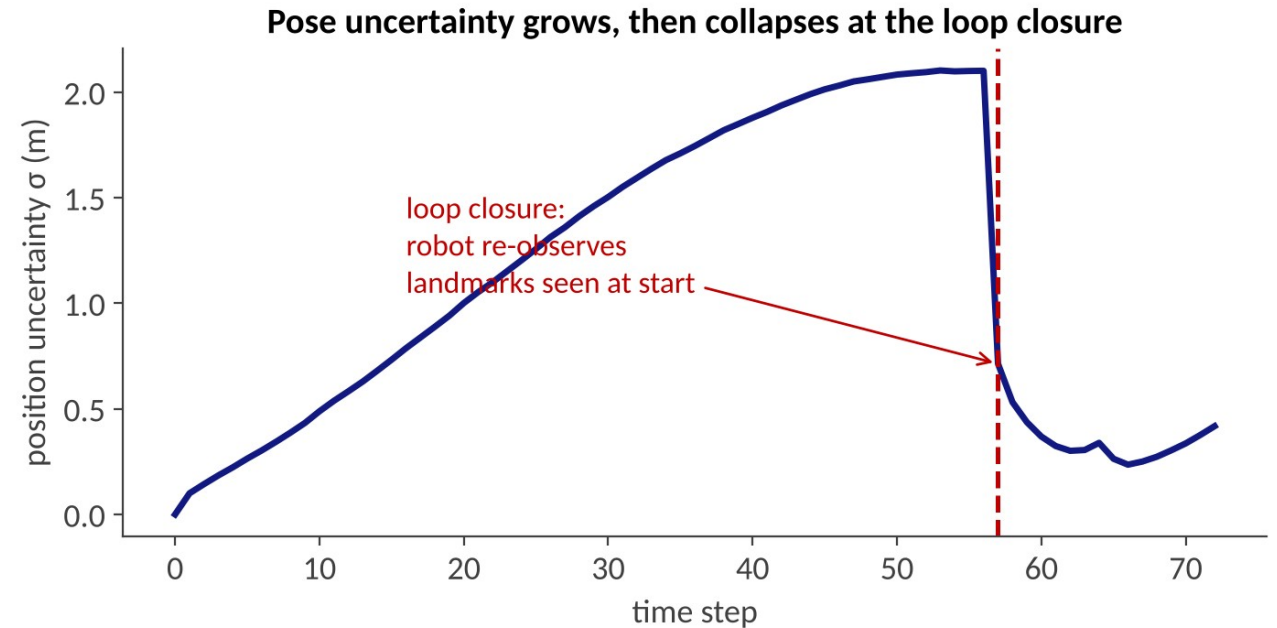
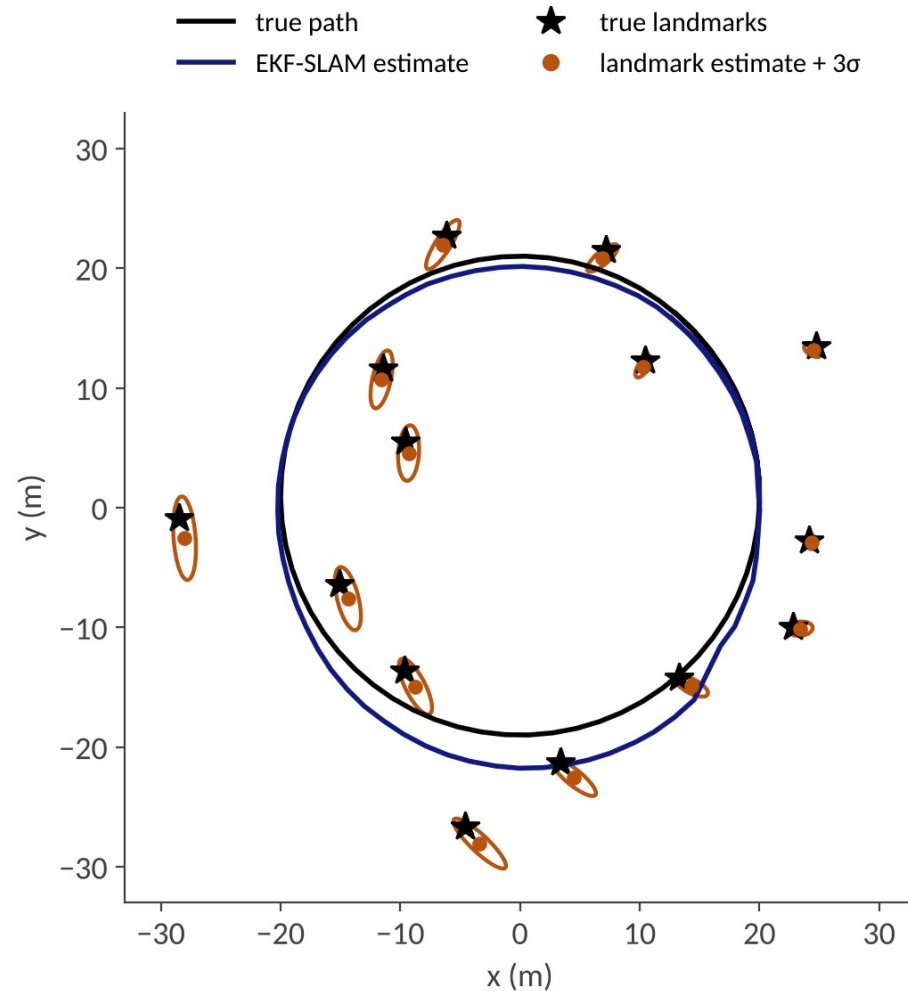
Correlations Are the Key

- Every observation links the robot and a landmark; over time **all** landmarks become correlated through the robot pose
- Correlations make the filter consistent: a correction for one landmark propagates to the others
- They also make Σ **dense**: $O(N^2)$ memory and update time
- Ignoring correlations (independent per-landmark filters) is overconfident and inconsistent



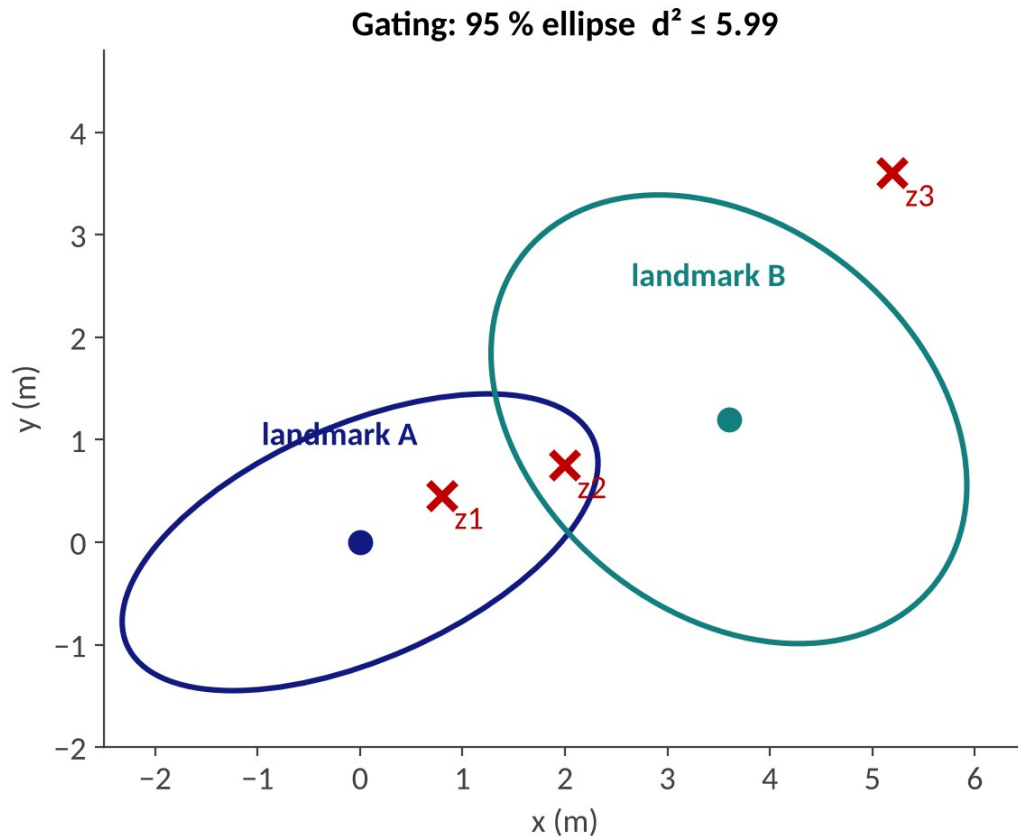
Left: after 12 steps (4 landmarks). Right: after 72 steps (14 landmarks). Dark = strong correlation.

EKF-SLAM in Action



- Pose σ grows to **2.1 m** while exploring, then drops to **0.4 m** at the loop closure ($t = 57$)
- The estimate "snaps back": loop closure corrects past drift in the map as well

Data Association and Gating

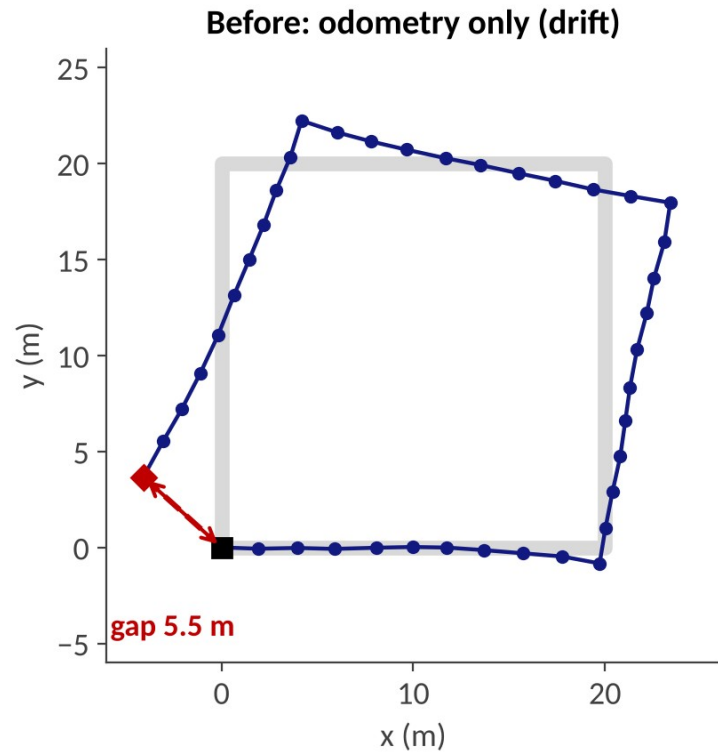


- Which landmark did this measurement come from?
Unknown in practice
- Gate by **Mahalanobis distance** $d^2 = v^T S^{-1} v$; for 2 DOF a 95 % gate is $d^2 \leq 5.99$
- $z_1 \rightarrow A$ only ($d^2 = 0.85$); z_2 is inside **both** gates (4.47 and 3.94): ambiguous; z_3 is outside all gates $\rightarrow$ new landmark or outlier
- Nearest neighbor is cheap; JCBB or multi-hypothesis methods handle ambiguity
- **A wrong match is catastrophic for EKF-SLAM**: it commits to one hypothesis and cannot undo it

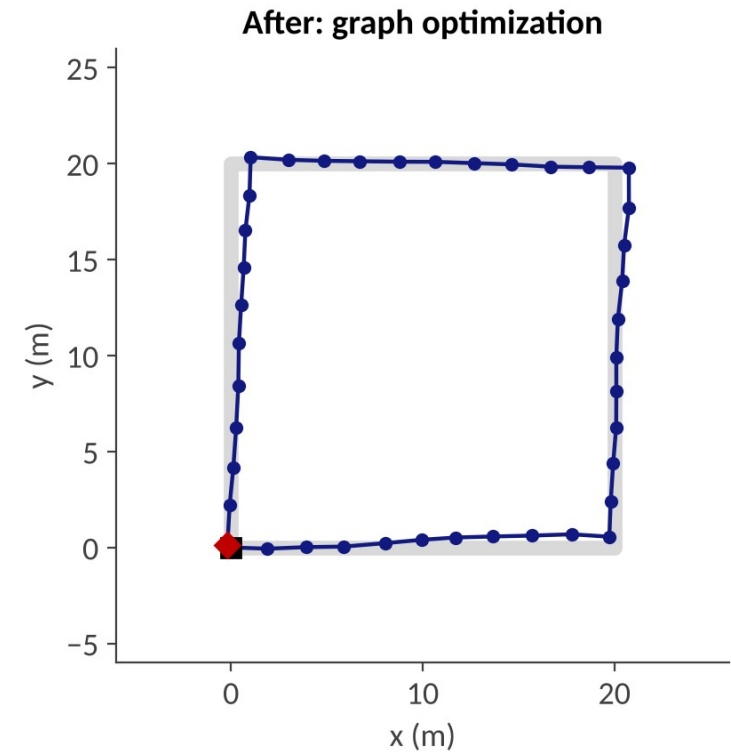
Limits of EKF-SLAM

- **Complexity:** $O(N^2)$ update and memory; thousands of landmarks are impractical
- **Linearization:** errors in the Jacobians make the filter overconfident (inconsistent), especially after large heading errors
- **Gaussian assumption:** cannot represent multi-modal pose belief, e.g., symmetric corridors
- **Single data-association hypothesis** per update (see previous slide)
- **Past poses are marginalized out:** cannot be re-linearized once a loop closure reveals a better estimate
- Still the best way to learn SLAM, and the core of many sensor-fusion filters

Part 3: Particle and Graph SLAM



— true path • pose nodes (odometry edges) - - loop-closure edge



FastSLAM | pose graphs | Gauss-Newton | sparsity

FastSLAM: Key Idea

- **Observation:** if the robot path were known, the landmarks would be **independent**
- So factor the posterior into a path term and one small term per landmark (Rao-Blackwellization):

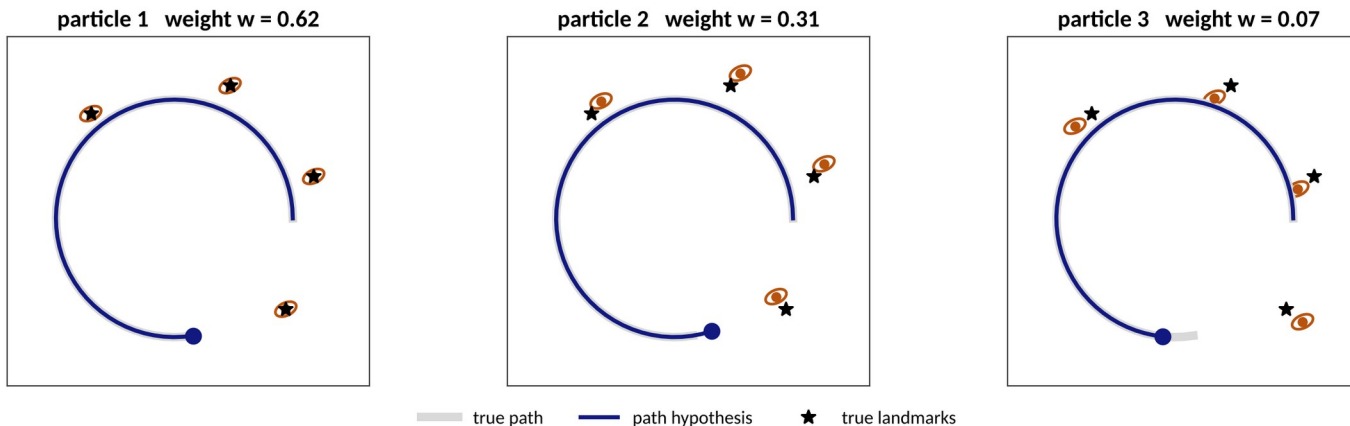
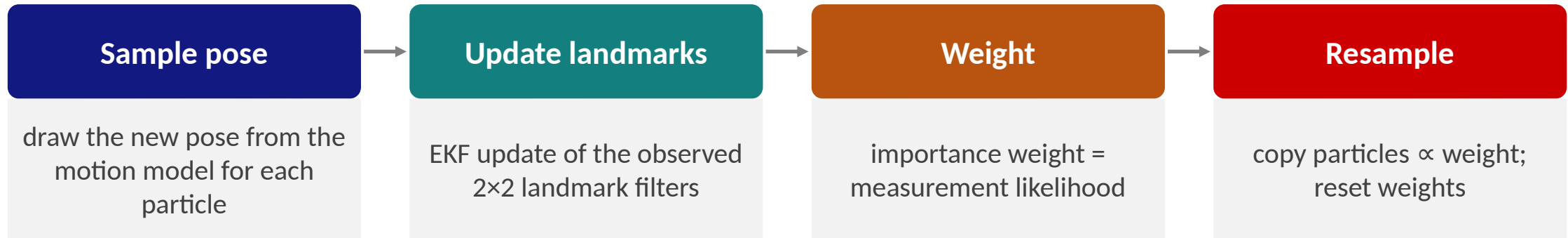
$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) = p(x_{1:t} \mid z_{1:t}, u_{1:t}) \prod_{j=1}^N p(m_j \mid x_{1:t}, z_{1:t})$$

↑ **path: particle filter**

↑ **one 2×2 EKF per landmark**

- Each **particle** = one path hypothesis + N tiny 2×2 EKFs (one per landmark), instead of one huge (3+2N) Gaussian
- Cost per observation is O(log N) per particle with a tree of landmarks: **O(M log N)** for M particles, versus O(N²) for EKF-SLAM
- Data association is decided **per particle**, so different particles can hold different hypotheses

FastSLAM: Algorithm and Behavior

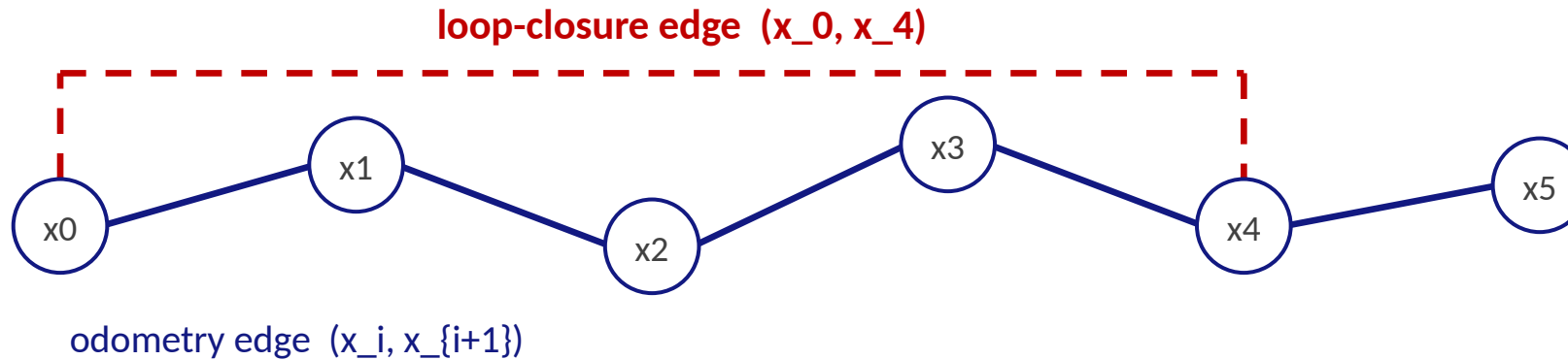


- Our simulation: $M = 40$ particles; after the loop closure the effective sample size fell to 6.8
- **Particle depletion:** resampling keeps only a few ancestors, so good hypotheses can be lost
- FastSLAM 2.0 samples the pose using the measurement as well

From Landmarks to Grids: GMapping

- Same idea with a **2D LiDAR** and an **occupancy grid**: every particle carries its own grid map
- Scan matching against the particle's map improves the proposal, so few particles are needed (tens)
- Widely used in ROS for indoor robots; memory grows with particles $\times$ map size
- Weak in large outdoor loops and in 3D: the reason graph-based methods took over

Graph SLAM: Nodes and Edges



- **Node** = a pose x_i (or a landmark); **edge** = a relative measurement z_{ij} between two nodes, with an information matrix Ω (the inverse of its covariance)
- Odometry edges connect consecutive poses; **loop-closure edges** connect distant poses that saw the same place
- No filtering: build the whole graph, then find the poses that best satisfy **all** constraints

The Graph SLAM Objective

Each edge predicts the relative pose from the current node estimates and compares it with the measurement z_{ij} :

$$e_{ij}(x) = h(x_i, x_j) - z_{ij}$$

$$x^* = \arg \min_x F(x) = \sum_{\langle i, j \rangle} e_{ij}^T \Omega_{ij} e_{ij}$$

- $h(x_i, x_j)$ = pose of node j seen from node i (in SE(2): a rotation and a translation); for 1-D it is just $x_j - x_i$
- Ω_{ij} **weights** each edge: precise edges (large Ω) pull harder than noisy ones
- This is a **nonlinear least-squares** problem: it is the maximum-likelihood estimate under Gaussian noise
- The same formulation is used for visual SLAM (bundle adjustment) and factor graphs

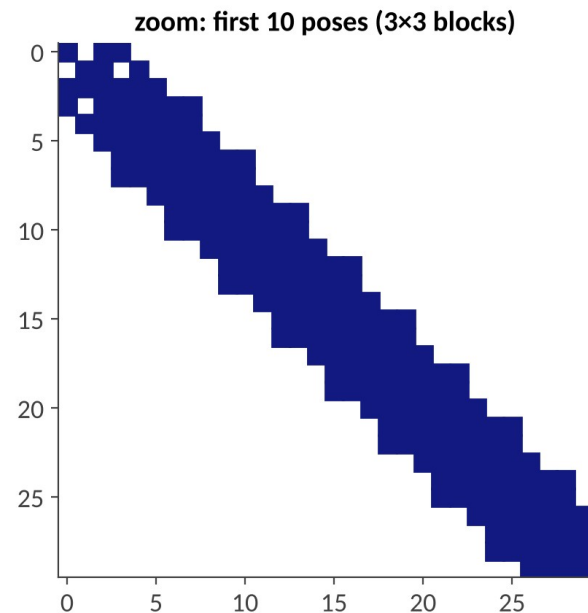
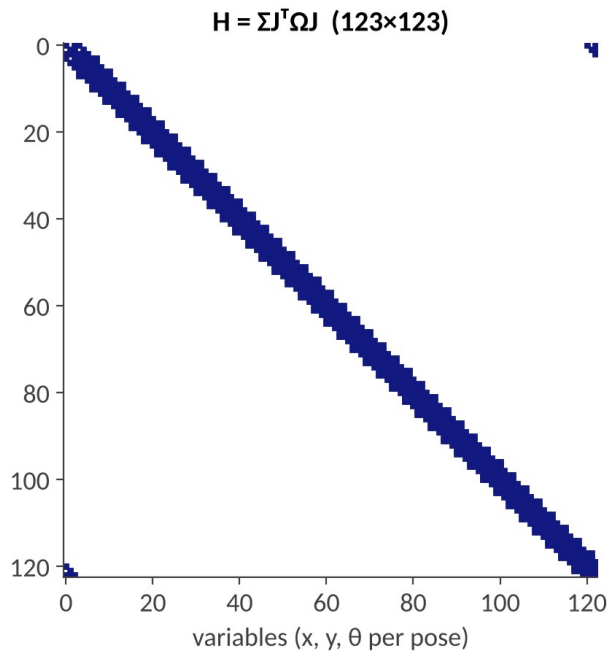
Solving It: Gauss-Newton

- Linearize each error around the current estimate: $e(x + \Delta x) \approx e + J \Delta x$
- Set the derivative of F to zero to get the **normal equations**; solve, update, repeat:

$$H \Delta x = -b, \quad H = \sum_{ij} J_{ij}^T \Omega_{ij} J_{ij}, \quad b = \sum_{ij} J_{ij}^T \Omega_{ij} e_{ij}, \quad x \leftarrow x + \Delta x$$

- Each edge adds a small contribution to H and b: only the blocks of its two nodes
- Iterate until Δx is small (our 41-pose example converges in about 6 iterations)
- Levenberg-Marquardt adds a damping term λI for robustness; g2o, GTSAM, and Ceres implement this
- Needs a **good initial guess** (usually odometry): the problem is nonconvex

Sparsity Makes It Fast



- 41 poses $\times$ 3 = **123×123** matrix, but only **6** % of entries are nonzero
- An edge touches only two nodes, so H mirrors the **adjacency of the graph**
- Odometry edges give the band; the loop-closure edge adds off-band blocks in the corners
- Sparse Cholesky solves large graphs (10^5 nodes and more) in seconds

Gauge freedom: relative constraints cannot fix the absolute pose, so H is singular. Fix one node (prior on x_0) or remove its rows and columns. H^{-1} then gives the covariance.

Worked Example: 1-D Pose Graph

Fixed $x_0 = 0$, unknowns x_1, x_2 . Odometry: $x_1 - x_0 = 1.0$ and $x_2 - x_1 = 1.0$ ($\sigma = 0.1$). "Loop-closure" edge: $x_2 - x_0 = 1.8$ ($\sigma = 0.05$). Start from the odometry guess $x = (1, 2)$.

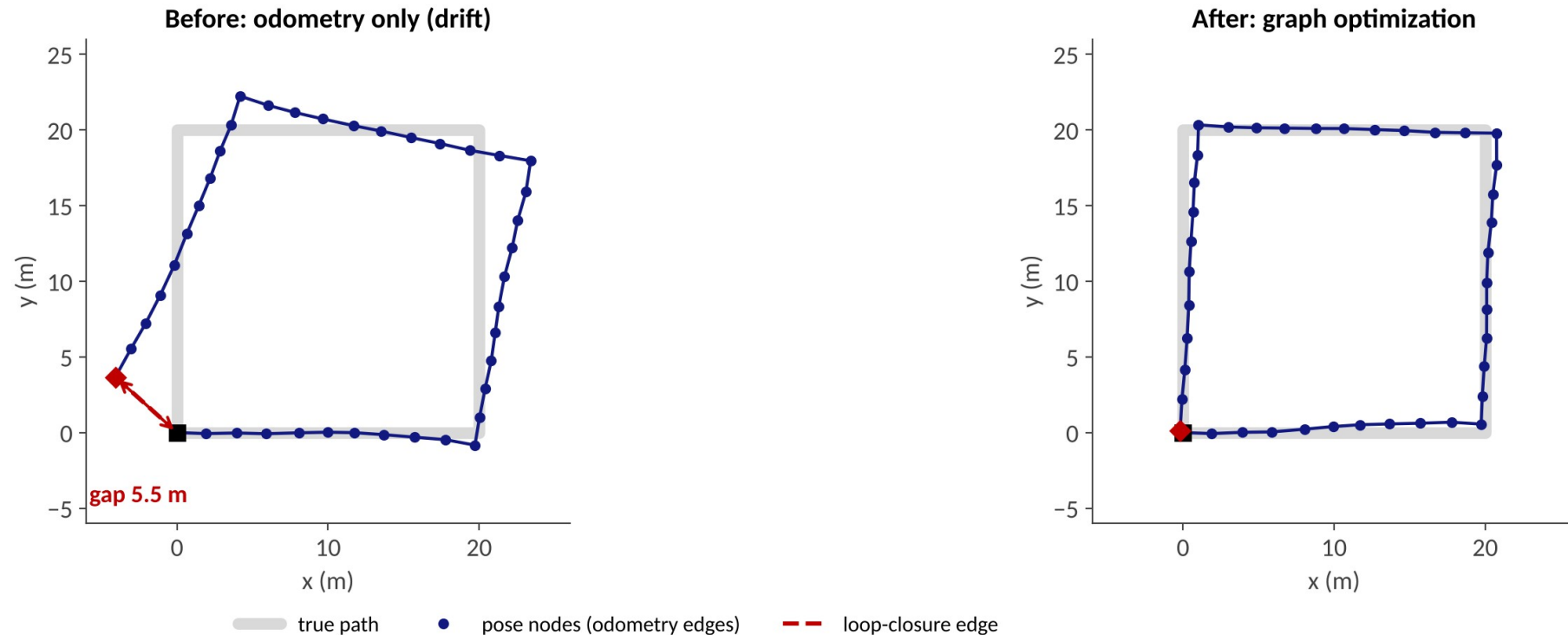
Edge	$\Omega = 1/\sigma^2$	e at $x = (1, 2)$	J (w.r.t. x_1, x_2)
$x_1 - x_0 = 1.0$	100	0	(1, 0)
$x_2 - x_1 = 1.0$	100	0	(-1, 1)
$x_2 - x_0 = 1.8$	400	+0.2	(0, 1)

$$H = \begin{bmatrix} 200 & -100 \\ -100 & 500 \end{bmatrix}$$

$$b = (0, 80), \quad H \Delta x = -b \rightarrow \Delta x = (-0.089, -0.178)$$

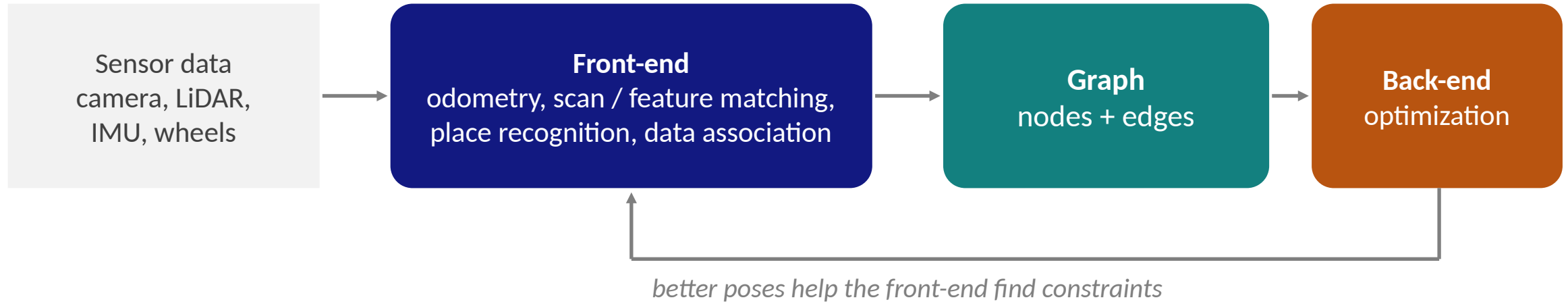
- Result: $x = (0.911, 1.822)$; the 0.2 m disagreement is spread over both edges, more on the less precise odometry
- H^{-1} gives the posterior covariance: $\sigma(x_1) \approx 0.075$, $\sigma(x_2) \approx 0.047$, tighter than the odometry alone
- The problem is linear, so one Gauss-Newton step is exact; checked against a direct solver

Pose Graph Optimization in Action



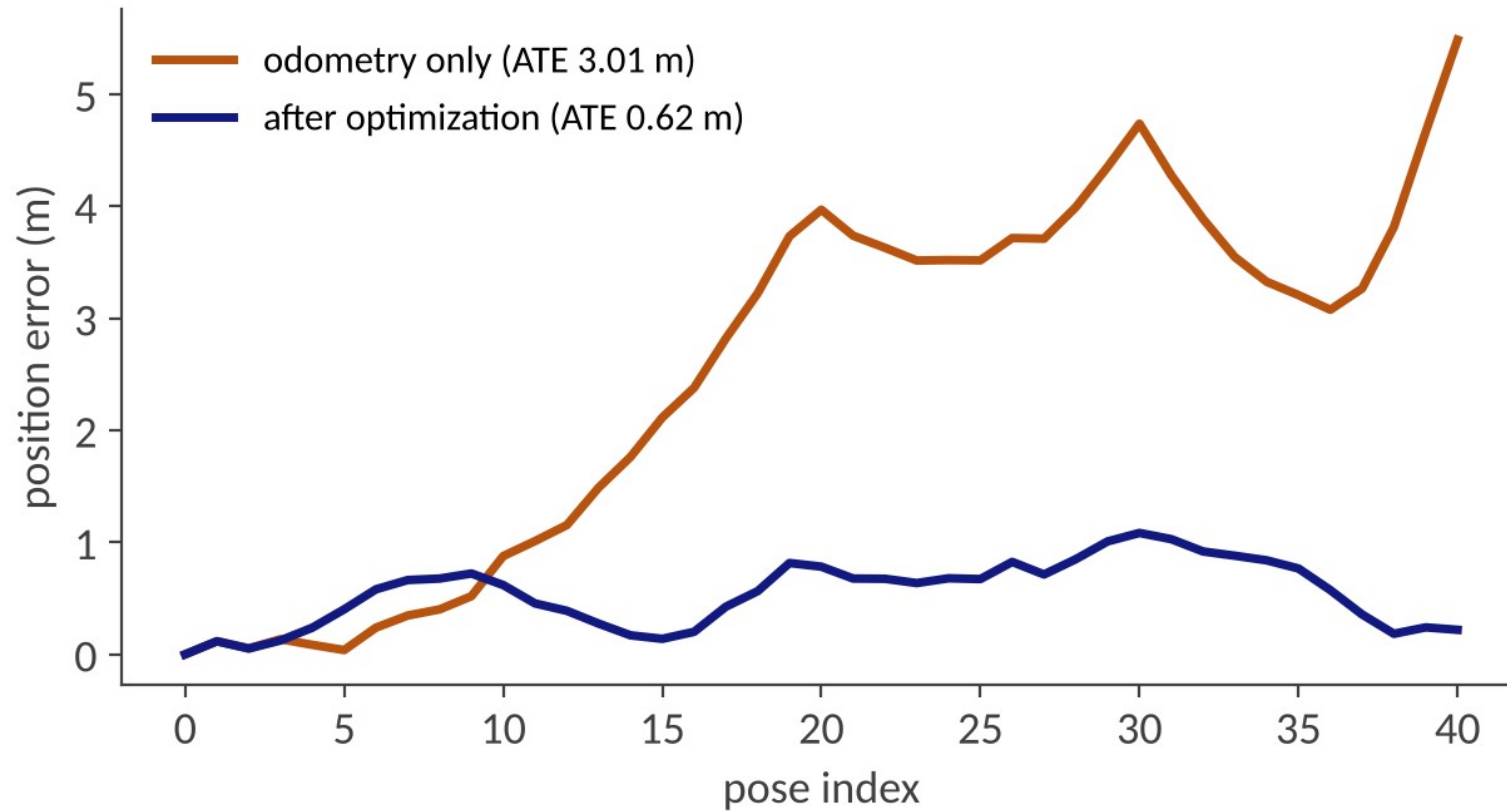
- 41 poses on a 20 m × 20 m square; odometry-only gap at the end **5.5 m**, after optimization **0.22 m**
- Trajectory RMSE (ATE): **3.01 m** → **0.62 m**; one loop-closure edge corrects the whole path

Front-End and Back-End



- **Front-end** (sensor-specific): turns raw data into constraints; **this is where most failures happen** (bad matches, false loops)
- **Back-end** (sensor-agnostic): optimizes the graph; robust against some bad edges (next part)
- This split is the architecture of nearly every modern SLAM system, including ORB-SLAM, LIO-SAM, and Cartographer

Part 4: Loop Closure and Evaluation

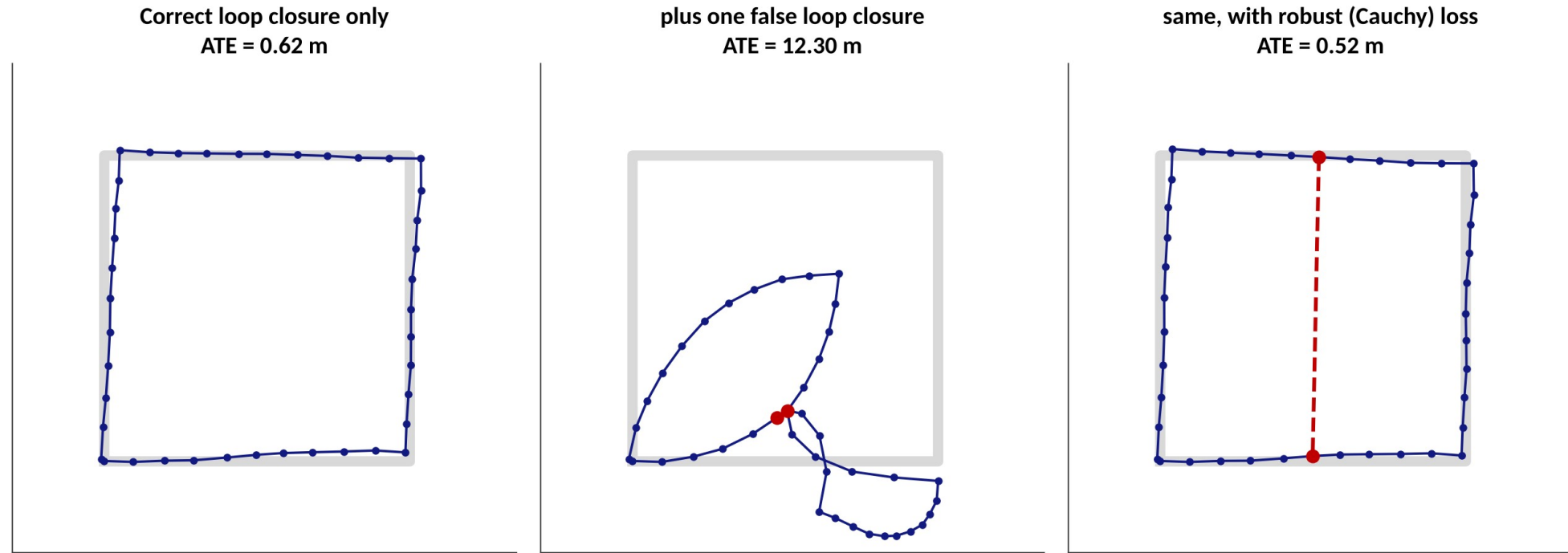


place recognition | false loops | robust back-ends | ATE and RPE

Loop Closure: Detect, Verify, Add

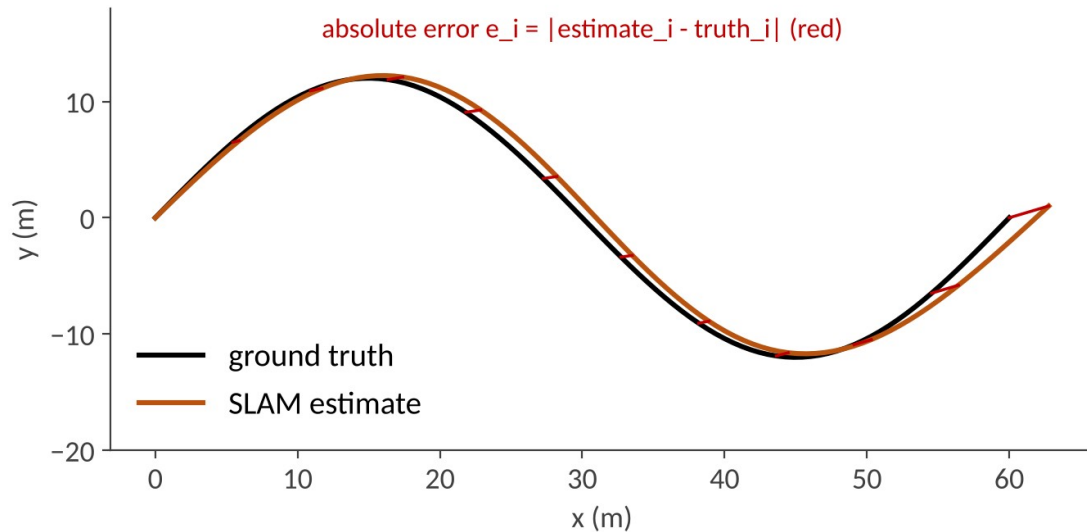
- **Detect:** find candidate places seen before; images → bag of visual words, LiDAR → global descriptors (e.g., Scan Context)
- **Verify:** geometric check (RANSAC + registration such as ICP) that the two views really align
- **Add:** insert an edge with the measured relative pose and its covariance; re-optimize
- Two failure types: **false negatives** (drift stays) and **false positives** (map is distorted)
- Perceptual aliasing: repeated structures (identical shelves, tunnels, highway segments) cause false positives

A False Loop Closure Ruins the Map



- A single false edge (poses 25 and 5 declared to be the same place) raises the ATE from 0.62 m to **12.3 m**
- A robust loss (Cauchy) **down-weights** the inconsistent edge: ATE 0.52 m. Other defenses: switchable constraints, consistency checks
- The robust run started from the previous estimate; robust losses do not remove the need for a good initial guess

Evaluating SLAM: ATE and RPE



$$\text{ATE}_{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n \|\hat{p}_i - p_i\|^2}$$

ATE (absolute trajectory error): global consistency; the estimate is first aligned to the ground truth (rigid or similarity transform).

RPE (relative pose error): drift over a fixed interval; compares relative motions.

- **KITTI odometry** reports translation error (%) and rotation error (deg/m) averaged over segments of 100 m to 800 m
- **EuRoC** and **TUM** report ATE RMSE in meters; always state whether scale was aligned (monocular systems)
- Evaluate with an established tool (e.g., the evo package) and report several runs: SLAM is stochastic

Which Method When?

	EKF-SLAM	FastSLAM	Graph SLAM
State kept	pose + N landmarks (Gaussian)	M paths + N·M small EKF's	all poses (and landmarks)
Cost per update	$O(N^2)$	$O(M \log N)$	sparse solve; incremental (iSAM2)
Revise the past?	no (marginalized)	only via resampling	yes (re-linearize)
Data association	one hypothesis	one per particle	can be verified and revoked
Sensor fusion	natural (filter)	awkward	natural (any factor)
Typical use today	fusion filters, teaching	2D grid mapping	visual and LiDAR SLAM, HD-map building

In-Class Activity and Lab Ideas

- **Hand calculation** (10 min): 1-D pose graph and 1-D EKF-SLAM from slides 21 and 34 with new numbers (see the exercise deck)
- **Python** (30 min): implement a 2-D pose graph with `scipy.optimize.least_squares`: odometry edges, one loop-closure edge, then add a false edge and a robust loss
- **Library demo**: run a GTSAM or g2o example dataset and plot the trajectory before and after optimization
- **Discussion**: if a loop closure is wrong by 5 m, which module (front-end or back-end) should detect it, and how?

Summary and Further Resources

Key takeaways

- SLAM estimates the **map and the pose together**; dead reckoning alone drifts without bound
- **EKF-SLAM**: one Gaussian, $O(N^2)$, landmarks become correlated; wrong data association is fatal
- **FastSLAM**: particles for the path, tiny EKF for landmarks, $O(M \log N)$
- **Graph SLAM**: poses and constraints, sparse Gauss-Newton; loop closures correct drift
- Evaluate with **ATE / RPE**; false loop closures are the main risk

Further resources

- Thrun, Burgard, Fox: **Probabilistic Robotics** (Ch. 10, 11, 13)
- Grisetti et al.: **A Tutorial on Graph-Based SLAM** (2010)
- Cadena et al.: **Past, Present, and Future of SLAM** (T-RO 2016)
- UW CSE 571 lecture slides (EKF-SLAM) and Univ. Bonn (C. Stachniss) graph-SLAM slides, both online
- GTSAM, g2o, and Ceres: open-source back-ends

References

- [1] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, 2005 (Ch. 10 EKF-SLAM, Ch. 11 GraphSLAM, Ch. 13 FastSLAM).
- [2] H. Durrant-Whyte, T. Bailey, "Simultaneous Localization and Mapping: Part I / Part II", IEEE Robotics & Automation Magazine, 2006.
- [3] R. Smith, M. Self, P. Cheeseman, "Estimating Uncertain Spatial Relationships in Robotics", 1990.
- [4] G. Grisetti, R. Kümmmerle, C. Stachniss, W. Burgard, "A Tutorial on Graph-Based SLAM", IEEE ITS Magazine, 2010.
- [5] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, "FastSLAM: A Factored Solution to the SLAM Problem", AAAI 2002.
- [6] G. Grisetti, C. Stachniss, W. Burgard, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters", IEEE T-RO, 2007.
- [7] C. Cadena et al., "Past, Present, and Future of SLAM", IEEE T-RO, 2016.
- [8] R. Kümmmerle et al., "g2o: A General Framework for Graph Optimization", ICRA 2011; F. Dellaert, M. Kaess, "Factor Graphs for Robot Perception", 2017.
- [9] J. Sturm et al., "A Benchmark for the Evaluation of RGB-D SLAM Systems", IROS 2012; A. Geiger et al., KITTI, CVPR 2012.
- [10] UW CSE 571 (2023), Lecture 8 EKF-SLAM: courses.cs.washington.edu/courses/cse571/23sp/slides/L08/Lecture08_EKFSLAM.pdf
- [11] C. Stachniss, Univ. Bonn, Graph-Based SLAM slides: ipb.uni-bonn.de/html/teaching/msr2-2020/sse2-05-graph-slam.pdf
- [12] Figures: original simulations created for this lecture (NumPy / SciPy, Matplotlib).