AprilTag-based Trajectory Tracking

for the LIMOS Robot

Joao Pedro Martins do Lago Reis and Yann Kelvem da Silva Ramos

Master's degree in artificial perception and robotics

December 9, 2025

Outline

Problem

Challenges

Project goal

Phases

Problem description

Problem to solve

- We need a robot that can **learn a path by being guided**.

Problem description

Problem to solve

- We need a robot that can **learn a path by being guided**.
- Later it must **drive that same path alone**.

Problem description

Problem to solve

- We need a robot that can **learn a path by being guided**.
- Later it must **drive that same path alone**.
- It has to work in normal indoor rooms without fuss.

Challenges

Why this is challenging

- **Hardcoded paths**: someone types coordinates by hand.

Challenges

Why this is challenging

- **Hardcoded paths**: someone types coordinates by hand.
- **No flexibility**: any small change means rebuilding code.

Challenges

Why this is challenging

- **Hardcoded paths**: someone types coordinates by hand.
- **No flexibility**: any small change means rebuilding code.
- **High barrier**: only experts can tweak routes.

Chosen setup

Selected Platform

- Robot: **LIMOS** with wheels and camera.

Fig. 1 — LIMOS mobile robot

Chosen setup

Selected Platform

- Robot: **LIMOS** with wheels and camera.
- Tags: **AprilTag** to tell where the robot is.

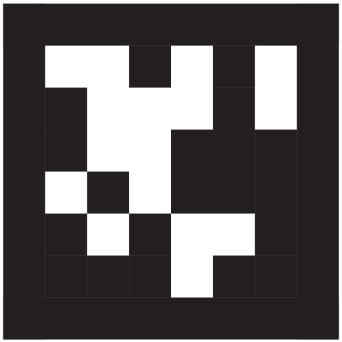


Fig. 2 — AprilTag visual marker

Chosen setup

Selected Platform

- Robot: **LIMOS** with wheels and camera.
- Tags: **AprilTag** to tell where the robot is.
- Together they keep the pose steady while we teach and repeat.

Fig. 1 — LIMOS robot

Fig. 2 — AprilTag marker

Project Goal

Our Goal

- Build a simple **teach and repeat** loop.

Project Goal

Our Goal

- Build a simple **teach and repeat** loop.

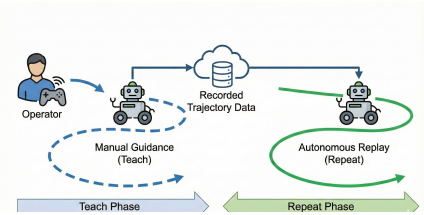


Fig.3 — High-level teach and repeat concept

Project Goal

Our Goal

- Build a simple **teach and repeat** loop.
- **No coding**: you just guide the robot by hand.

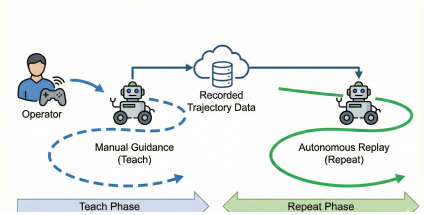


Fig.3 — High-level teach and repeat concept

Project Goal

Our Goal

- Build a simple **teach and repeat** loop.
- **No coding**: you just guide the robot by hand.
- **Auto replay**: it drives the same path accurately later.

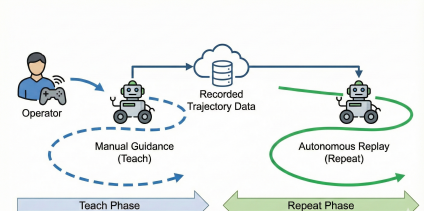


Fig.3 — High-level teach and repeat concept

Phase 1: TF setup and perception

Step 1 — Put tags on the map.

We save where each AprilTag sits:

$$T_{\text{world}}^{\text{tag}}$$

This stays fixed for every next step.

Phase 1: TF setup and perception

Step 2 — Drive the robot once.

The simulator gives the robot pose while it follows a smooth curve:

$$T_{\text{world}}^{\text{camera\_link}}(t)$$

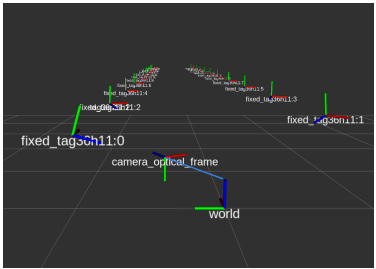


Fig. 4 — World, robot and camera frames

Phase 1: TF setup and perception

Step 3 — Use the real camera frame.

We switch to the optical frame (how the camera really looks):

$$T^{\text{camera\_optical}}_{\text{world}}(t) = T^{\text{camera\_link}}_{\text{world}}(t) \cdot T^{\text{camera\_optical}}_{\text{camera\_link}}$$

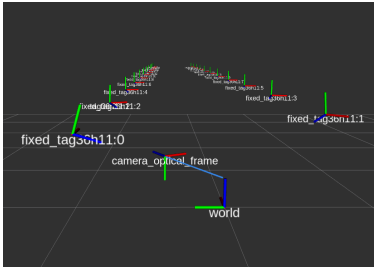


Fig. 4 — World, robot and camera frames

Phase 1: TF setup and perception

Step 4 — Compute clean tag detections.

From the map and pose we get, for each tag:

$$T^{\text{tag}}_{\text{camera\_optical}}(t)$$

This is the clean data that Phase 2 will dirty with noise.

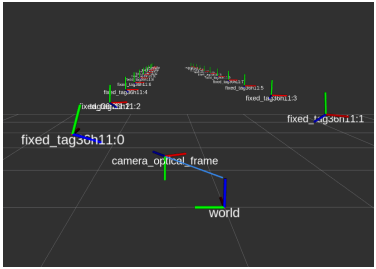


Fig. 4 — World, robot and camera frames

Phase 2: Simulation and dataset generation

Step 1 — Simulate a clean path.

We make a smooth curve and log the true pose:

$$T_{\text{world}}^{\text{camera\_optical}}(t)$$

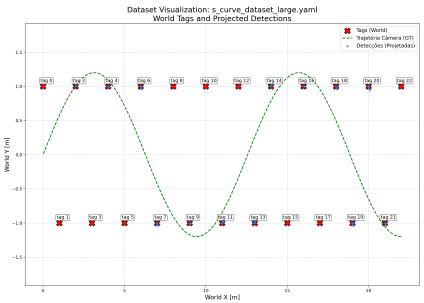


Fig. 5 — Ground truth sinusoid trajectory

Phase 2: Simulation and dataset generation

Step 2 — Compute perfect tag reads.

For every tag and time we compute the clean detection:

$$T^{\text{tag}}_{\text{camera\_optical}}(t)$$

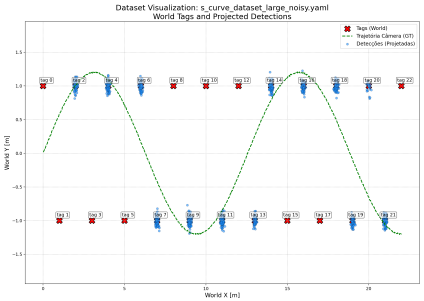
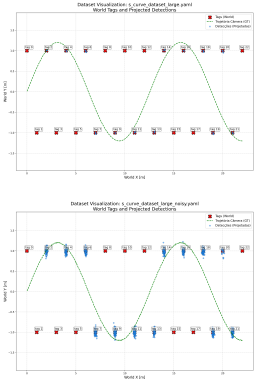


Fig. 6 — Noisy measurements

Phase 2: Simulation and dataset generation

Step 3 — Add noise like real life.

We add Gaussian noise so readings look imperfect, like sensors do.



Clean (top) and noisy (bottom) simulated data

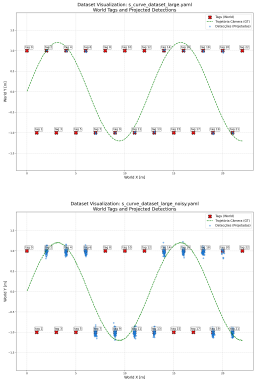
Phase 2: Simulation and dataset generation

Step 4 — Save the dataset.

For each time we save:

- the true pose $T_{\text{world}}^{\text{camera\_optical}}(t)$
- the noisy tag detections $T_{\text{camera\_optical}}^{\text{tag}}(t)$

Phase 3 will test its optimizer on this file.



Clean (top) and noisy (bottom) simulated data

Phase 3: Trajectory Optimization & Recovery

Goal: Fix drift with a simple optimizer

- **Path recovery:** Align the noisy run with the taught path.

Phase 3: Trajectory Optimization & Recovery

Goal: Fix drift with a simple optimizer

- **Path recovery:** Align the noisy run with the taught path.
- **Pose graph:** Reduce error between waypoints and simple constraints.

Phase 3: Trajectory Optimization & Recovery

Goal: Fix drift with a simple optimizer

- **Path recovery:** Align the noisy run with the taught path.
- **Pose graph:** Reduce error between waypoints and simple constraints.
- **Drift fix:** Auto-correct rotation and position drift.

Recovery the Optimal Path Sinusoidal Trajectory

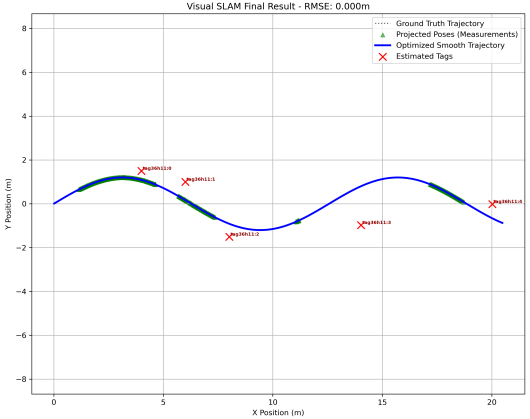


Fig. 7.2 — Trajectory Recovery via Optimization

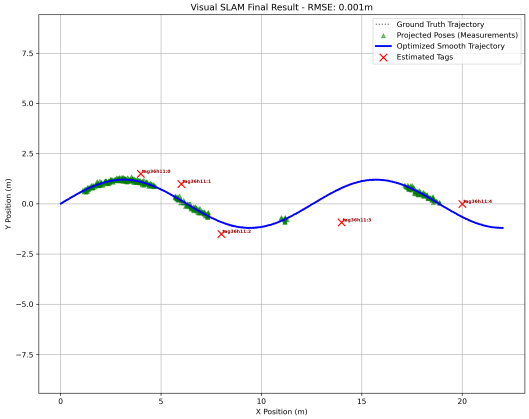


Fig. 7.1 — Simulated Dataset with Gaussian Noise

Quantitative Analysis Sinusoidal Trajectory

Case 1: Optimization without NOISE (Ideal)

Metric	Value
Trajectory RMSE	**0.0000** m
Avg. Tag Position Error	0.0164 m

Tag Estimation Error (Ideal)

Tag Name	X Error	Y Error	Z Error
tag36h11:0	0.00	0.00	0.00
tag36h11:1	0.00	0.00	0.00
tag36h11:2	0.01	−0.01	0.00
tag36h11:3	0.03	0.00	0.00
tag36h11:4	0.02	−0.02	0.00

Case 2: Optimization with NOISE (6 cm)

Metric	Value
Trajectory RMSE	**0.0007** m
Avg. Tag Position Error	0.0239 m

Tag Estimation Error

Tag Name	X Error	Y Error	Z Error
tag36h11:0	0.00	0.01	0.01
tag36h11:1	0.01	0.00	0.00
tag36h11:2	0.00	−0.04	0.01
tag36h11:3	−0.01	0.07	0.00
tag36h11:4	0.00	−0.01	0.00

Tables — Comparison of Optimization Results with and without noise.

Recovery the Optimal Path Sinusoidal Trajectory

The scenario with 23 tags

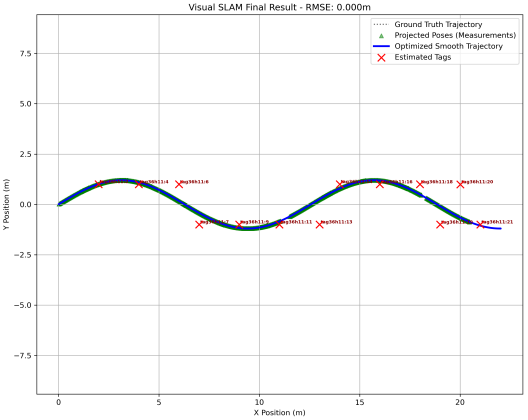


Fig. 8.1 — Trajectory Recovery via Optimization

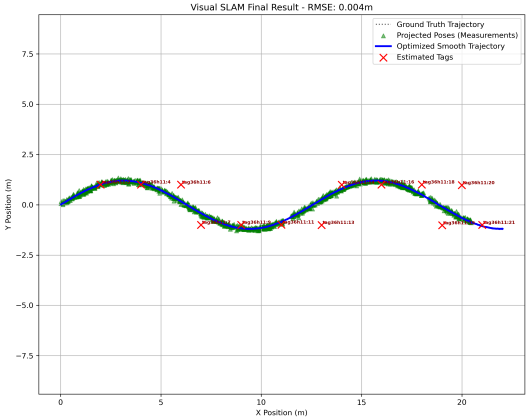


Fig. 8.2 — Simulated Dataset with Gaussian Noise

Quantitative Analysis Sinusoidal Trajectory

The scenario with 23 tags

Case 1: Optimization without NOISE

Metric	Value
Trajectory RMSE	**0.0002** m
Avg. Tag Position Error	0.0019 m

Tag Estimation Error (Sample)

Tag Name	X Error	Y Error	Z Error
tag36h11:2	0.00	0.00	0.00
tag36h11:7	0.00	0.00	0.00
tag36h11:13	0.00	0.00	0.00
tag36h11:18	0.00	0.00	0.00
tag36h11:21	0.00	0.00	0.00

Case 2: Optimization with NOISE (6 cm)

Metric	Value
Trajectory RMSE	**0.0037** m
Avg. Tag Position Error	0.0134 m

Tag Estimation Error (Sample)

Tag Name	X Error	Y Error	Z Error
tag36h11:2	0.01	0.01	0.00
tag36h11:7	0.00	0.00	0.00
tag36h11:13	0.01	0.01	0.00
tag36h11:18	0.00	0.00	0.01
tag36h11:21	0.01	0.00	0.02

Tables — Comparison of Optimization Results with and without noise (Sample of 5 Tags).

Recovery the Optimal Path: Straight Trajectory

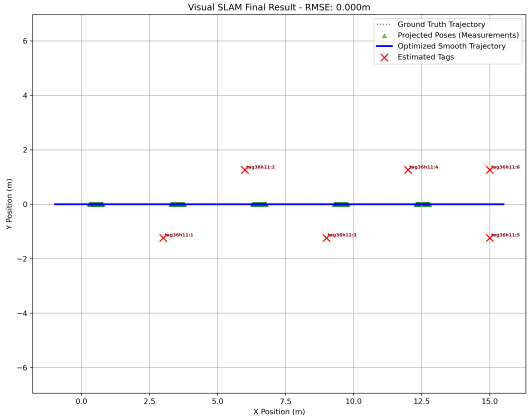


Fig. 8.1 — Trajectory Recovery via Optimization

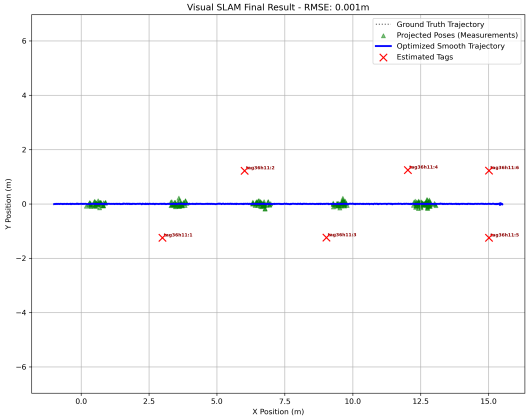


Fig. 8.2 — Simulated Dataset with Gaussian Noise

Quantitative Analysis: Straight Trajectory

Case 1: Optimization without NOISE

Metric	Value
Trajectory RMSE	**0.0017** m
Avg. Tag Position Error	0.0000 m

Tag Estimation Error (Sample)

Tag Name	X Error	Y Error	Z Error
tag36h11:0	0.00	0.00	0.00
tag36h11:1	0.00	0.00	0.00
tag36h11:2	0.00	0.00	0.00
tag36h11:3	0.00	0.00	0.00
tag36h11:4	0.00	0.00	0.00

Case 2: Optimization with NOISE (6 cm)

Metric	Value
Trajectory RMSE	**0.017** m
Avg. Tag Position Error	0.01912 m

Tag Estimation Error (Sample)

Tag Name	X Error	Y Error	Z Error
tag36h11:0	0.01	0.01	0.00
tag36h11:1	0.00	0.00	0.00
tag36h11:2	0.01	0.01	0.00
tag36h11:3	0.00	0.00	0.01
tag36h11:4	0.01	0.00	0.02

Tables — Comparison of Optimization Results with and without noise (Sample of 5 Tags).

Phase 5: Path recovery algorithm

Goal: Recover when the robot leaves the taught path

- **Recovery logic:** Simple rules to steer back to the recorded path.

Phase 5: Path recovery algorithm

Goal: Recover when the robot leaves the taught path

- **Recovery logic:** Simple rules to steer back to the recorded path.
- **Closest waypoint search:** Use a quick KD-Tree to find the nearest waypoint.

Phase 5: Path recovery algorithm

Goal: Recover when the robot leaves the taught path

- **Recovery logic:** Simple rules to steer back to the recorded path.
- **Closest waypoint search:** Use a quick KD-Tree to find the nearest waypoint.

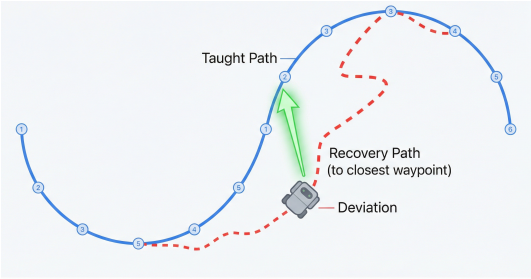


Fig. 10 — Returning to the original path after deviation

Phase 5: Path recovery algorithm

Goal: Recover when the robot leaves the taught path

- **Recovery logic:** Simple rules to steer back to the recorded path.
- **Closest waypoint search:** Use a quick KD-Tree to find the nearest waypoint.
- **Safety validation:** Test recovery moves in sim to avoid bad turns.

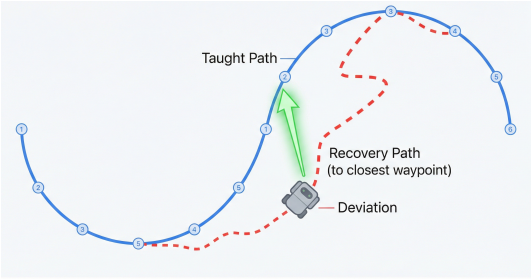


Fig. 10 — Returning to the original path after deviation

Phase 6: System integration

Goal: Demonstrate full teach and repeat autonomy

- **Controller integration:** Mix Pure Pursuit with the perception and pose stack.

Phase 6: System integration

Goal: Demonstrate full teach and repeat autonomy

- **Controller integration:** Mix Pure Pursuit with the perception and pose stack.
- **Autonomous replay:** Let the robot drive the recorded path on its own.

Phase 6: System integration

Goal: Demonstrate full teach and repeat autonomy

- **Controller integration:** Mix Pure Pursuit with the perception and pose stack.
- **Autonomous replay:** Let the robot drive the recorded path on its own.

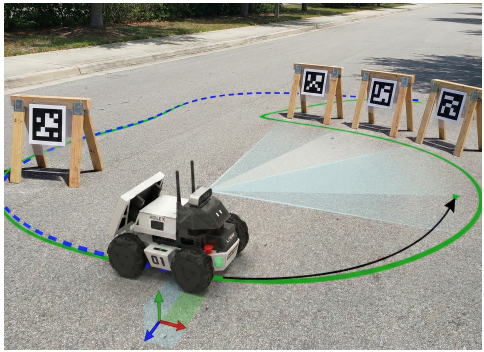


Fig. 11 — Fully integrated teach and repeat behavior

Phase 6: System integration

Goal: Demonstrate full teach and repeat autonomy

- **Controller integration:** Mix Pure Pursuit with the perception and pose stack.
- **Autonomous replay:** Let the robot drive the recorded path on its own.
- **Final system:** A teach & repeat pipeline anyone can run without coding.

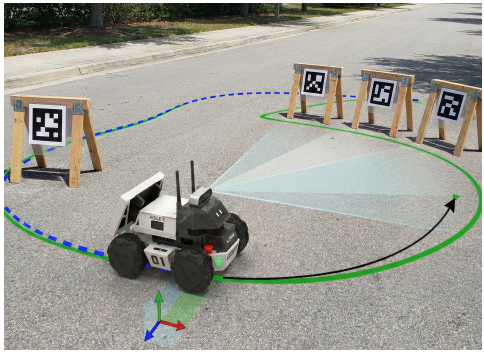


Fig. 11 — Fully integrated teach and repeat behavior

References I

[1] R. Adámek, M. Brablc, P. Vávra, B. Dobossy, M. Formánek, and F. Radil, “Analytical Models for Pose Estimate Variance of Planar Fiducial Markers for Mobile Robot Localisation”, *Sensors*, vol. 23, no. 12, p. 5746, 2023. DOI: 10.3390/s23125746

[2] B. S. Herrera, M. García, W. Chamorro, and D. Maldonado, “Visual Mapping and Autonomous Navigation Using AprilTags in Omnidirectional Mobile Robots: A Realistic ROS-Gazebo Simulation Framework”, *Engineering Proceedings*, vol. 115, no. 1, p. 5, 2025. DOI: 10.3390/engproc2025115005

[3] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2005. DOI: 10.5555/1625995.1626105

[4] B. Pfrommer and K. Daniilidis, “TagSLAM: Robust SLAM with Fiducial Markers”, *arXiv preprint arXiv:1910.00679*, 2019. DOI: 10.48550/arXiv.1910.00679

Thank you for listening!

Any questions?

Joao Pedro Martins do Lago Reis | Yann Kelvem da Silva Ramos

AprilTag-based Trajectory Tracking for the LIMOS Robot