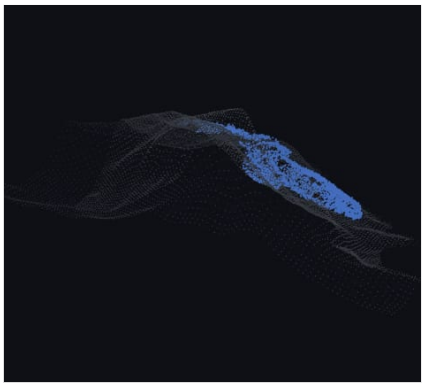
PORTFOLIO · OCTOBER 2026

Scan. Model. Decipher.

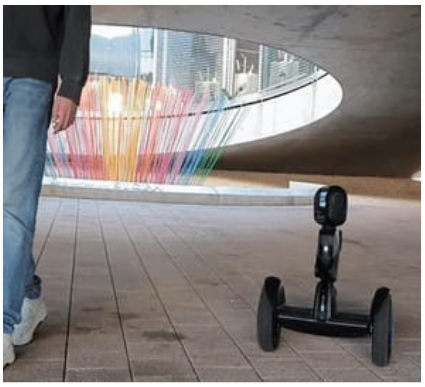
Making hidden structure visible.

Research Fellow, MIT Senseable City Lab Rio · MSc Robotics, EPFL
Research: how city form shapes light and air.
Practice: making autonomous systems legible.



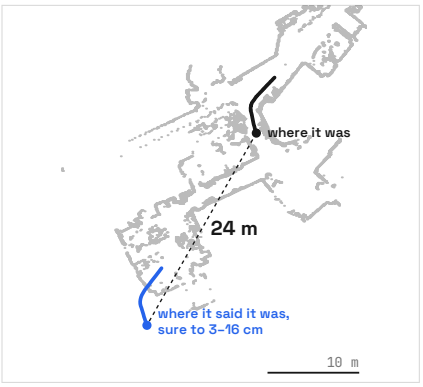
FAVELA RESEARCH

Vidigal, Rio de Janeiro · terrain and building footprints



ROBOTICS

Segway Loomo, EPFL VITA Lab · 2022–23



PRACTICE

PR2, MIT Stata Center dataset (CC BY 3.0): in one corridor, sure it was in another

Théo Alessandro Hermann

thermann.ai@gmail.com · theoalessandro.com · theohermann.ch
github.com/thrmnn · linkedin.com/in/theohermann-epfl

ABOUT

I'm Théo, a French-Brazilian roboticist and urban scientist. Paris, São Paulo, Rio, Lausanne, Amsterdam, Barcelona, Stockholm.

I trained in robotics at EPFL, built perception for a person-following robot at VITA Lab and for an autonomous boat at Roboat, then brought the same tools to cities at MIT Senseable City Lab.

Wherever I work, the question is the same: what aren't we seeing yet that explains how the system behaves? Today I help robot makers find out why their systems fail, and I study how city form shapes light and air.

Speaking Dubai Future Week, 18–21 Nov 2026

IN THIS PORTFOLIO

FAVELA RESEARCH

How city form shapes light and air MIT Senseable City Lab Rio · since 2025 3

A street-level sensor anyone can build MIT Senseable City Lab Rio · Octopus 4

ROBOTICS

Autonomous mobile robot navigation on a Segway Loomo EPFL VITA Lab · 2022–23 5

Camera perception for canal boats Roboat, MIT spinoff · first intern, 2024 6

Event cameras for high-speed drone state estimation IRI (CSIC-UPC, Barcelona) · 2023–24 7

A real-time vision pipeline Open source · 2026 8

PRACTICE

Finding the moment a robot got lost Open source · 2026 9

A harness for AI agents Open source · 2026 10

MORE WORK

Leaf area from an aerial scan Senseable City Lab, MIT and AMS Institute · since 2024 11

FAVELA RESEARCH · MIT Senseable City Lab Rio · since 2025

How city form shapes light and air

Research fellow since January 2025, there from the Rio branch's first week

Hidden	How light and air move through a city built without plans, which measures made for street grids miss.
Revealed	3D models of the fabric, from the whole city down to a single alley, to study sunlight and airflow where people live.
Changed	Measures that hold on a fabric the standard ones were not built for.



Vidigal, Rio de Janeiro · terrain and building footprints

THE PROBLEM

Most ways of describing a city assume that someone drew it first: streets with names, plots with edges, blocks that repeat. Informal settlements are not built that way, so the standard descriptors either fail outright or quietly measure the wrong thing. Anything you want to say afterwards about how air, light or heat move through that fabric depends on describing its geometry first.

WHAT I BUILD

A Python pipeline that turns building footprints and a digital terrain model into a grid of morphometric indicators, fine enough that a single alley is resolved rather than averaged away. The indicators are the standard vocabulary of urban physics, computed so they hold on a fabric they were not designed for: sky openness, frontal area density, height variation, slope, aspect, porosity.

On top of that sits a sampling design that decides where a computational fluid dynamics campaign should actually look, rather than simulating everywhere at uniform cost, and an ingestion layer that turns public meteorological records into the boundary conditions those simulations need. The simulations themselves run elsewhere; this work defines what they receive and how their output is read back.

WHAT IS SHOWN HERE

The only picture of this work is the geometry of one site, Vidigal, on the front page of theoalessandro.com. Nothing else from this work appears: this page describes the method only, with no indicators, no findings and no figures from the study.

The full case: theoalessandro.com/projects/urban-morphometrics

Urban morphometrics, Point clouds, LiDAR, Terrain models, Python

theoalessandro.com/projects/urban-morphometrics

A street-level sensor anyone can build

Research fellow in the Rio lab: projects and deployment, research meetings, talks

Hidden	Heat, humidity and fine particles at street level, where people actually walk and live.
Revealed	A sensor anyone can build: an Arduino board, a fine-particle sensor and GPS in a 3D-printed case, assembled in about 20 minutes, logging a line every 5 seconds.
Changed	Street-level air and heat can be mapped without a lab's equipment; in Maré, the measurements are being linked to urban form.

WHAT IT IS

Octopus is MIT Senseable City Lab's open-source sensor kit for street-level heat, humidity and fine particles, the step after City Scanner (2017, on garbage trucks in Cambridge, Massachusetts) and Flatburn (2023). An Arduino board, a fine-particle sensor and a GPS receiver sit in a 3D-printed case. It assembles in about 20 minutes and writes one line of data every 5 seconds to a microSD card, first on vehicles, then anywhere.

CREDITS

Octopus is led by Simone Mora and Fabio Duarte, with Carlo Ratti; electronics by FAE Technology.

WHAT I DO

In the Rio lab I work on its projects and deployment, take part in the research meetings, and present the hardware (Casa Firjan summer school, October 2026). I am helping shape the new research line that links its measurements to urban form in Maré.

Environmental sensing, Open hardware, Arduino, GPS, 3D printing theoalessandro.com/projects/octopus

ROBOTICS · EPFL VITA Lab · 2022–23

Autonomous mobile robot navigation on a Segway Loomo

Research assistant, EPFL VITA Lab, under Alexandre Alahi · Lausanne · June 2022 to June 2023

Hidden	Where the people around a robot are and where they are heading, in a crowded indoor space.
Revealed	A modular ROS pipeline on a Segway Loomo: person detection and tracking, trajectory prediction, path planning and model predictive control. The base is C. Conejo Bernal's; I redid its architecture and its perception block.
Changed	Validated end to end on the physical robot, not only in simulation; the code is public. No figures were published: the evidence is the video in the repo.



Segway Loomo, EPFL VITA Lab · 2022–23

PROBLEM

Enable Segway Loomo robots to navigate crowded, dynamic indoor environments; most academic navigation systems never leave simulation.

BUILT

A modular ROS pipeline that runs on a server and drives the Loomo over a wireless socket link, one package per job:

- **Perception:** person detection (YOLOv5), a single-object tracker that holds on to the person being followed (Stark), and their 2D and 3D keypoints.
- **Estimation:** one node turns the IMU stream the Loomo sends into the robot's state, relative to where it started; another builds a map and estimates the state from all its sensors.
- **Prediction:** where the followed person is heading.
- **Planning and control:** a collision-free path, followed with model predictive control.

It keeps the base of C. Conejo Bernal's earlier pipeline (2020–21); I redid its architecture and its perception block in 2022–23, continuing the VITA Lab research line. The perception models and the ROS 1 workspace are pinned to the 2023 stack; the modern successor of the perception is the real-time vision pipeline.

WHY HARD

Real-time performance across several computationally intensive perception modules running simultaneously, validated among actual pedestrians, not just in simulation.

WHAT HAPPENED

Validated end-to-end on physical robots (Segway Loomo, EPFL VITA Lab research platform); the code is public on GitHub (`github.com/thrmnn/ROS_Autonomous_Driving`).

Path Planning, Autonomous Navigation, Multi-Object Tracking, ROS, C++, OpenCV, Mobile Robotics

theoalessandro.com/projects/autonomous-mobile-robot
github.com/thrmnn/ROS_Autonomous_Driving

Roboat, MIT spinoff · first intern, 2024

Camera perception for canal boats

First intern, on computer vision and MLOps · Amsterdam · March to August 2024

Hidden	An autonomous boat that read the canals through LiDAR alone.
Revealed	Camera detection and tracking, running on the boat's own edge computer.
Changed	A second sense on board, and every model traceable to the recordings it learned from.

THE PROBLEM

Roboat's autonomy already worked, and it worked on LiDAR. LiDAR is precise and expensive, and on water it reports a great deal that is not an obstacle: reflection, wake, chop. The open question when I arrived was how much of the perception job a camera could take on instead, on hardware that fits on a small vessel and inside a real-time budget.

WHAT I DID

I joined as the first intern and built the camera side of the stack: detection and multi-object tracking running on the vessel's own edge hardware, trained on the recordings the boats had been collecting from the canals. Water is an awkward subject for a detector, because the same object looks different under glare, reflection and weather, and the things you most need to catch are the things you have least footage of.

The second half of the work was the part that outlives any particular model. Datasets and model versions went under DVC so a result could be traced back to the exact data that produced it, and retraining stopped being a thing one person could do on one laptop.

Object Detection, Multi-Object Tracking, PyTorch, YOLO, ROS 2, Jetson Orin, Edge Deployment

theoalessandro.com/projects/autonomous-boat-perception

IRI (CSIC-UPC, Barcelona) · 2023–24

Event cameras for high-speed drone state estimation

Visiting research fellow, IRI (CSIC-UPC), with Juan Andrade-Cetto · Barcelona · October 2023 to February 2024

Hidden	Where a drone is during aggressive flight, once frame cameras blur and standard visual-inertial odometry diverges.
Revealed	A PyTorch network that estimates 6-DOF pose from event-camera, RGB and IMU streams, trained on high-speed drone flights.
Changed	A pose estimator aimed at flight above 5 g, the regime where frame-based pipelines fail; the training framework is open source. No accuracy figures were published.

PROBLEM

Conventional frame-based cameras suffer motion blur at high angular and linear velocity, causing standard visual-inertial odometry (VIO) to diverge during aggressive quadrotor maneuvers exceeding 5 g acceleration.

BUILT

A PyTorch network that estimates 6-DOF pose from three streams: an event camera (asynchronous per-pixel brightness changes, microsecond resolution), an RGB camera and an IMU.

- **Early fusion:** event and RGB frames are fused on a shared spatial tensor.
- **IMU encoder:** a WaveNet encoder reads the IMU stream over time.
- **Pose head:** a multi-modal head regresses the 6-DOF pose.

Trained on the UZH-FPV drone-racing dataset and the lab's own IRI-Borinot recordings (RGB, IMU and an event camera on a drone), with Juan Andrade-Cetto.

WHY HARD

Event cameras eliminate motion blur but require fundamentally different processing than standard frame-based VIO pipelines.

WHAT HAPPENED

The model targets maneuvers above 5 g, the regime where conventional frame cameras and standard VIO diverge. The training framework is open source (EVIO); real-time onboard inference lives in a separate repository that is not yet open.

State Estimation, Visual-Inertial Odometry, Sensor Fusion, Event Cameras, Deep Learning, PyTorch, ROS, C++

theoalessandro.com/projects/event-cameras-drone
github.com/thrmnn/EVIO

Open source · 2026

A real-time vision pipeline

Hidden — Four versions of the same detection and tracking work, built 2022 to 2023, sharing no interface and pinned to stacks that aged out.

Revealed — One modular stack behind narrow interfaces. Version 0.1 ships detection and tracking; re-identification and pose follow.

Changed — A module can be swapped without touching the rest, and every module reports its own latency. No benchmark is published: those timings depend on the machine. Open source, MIT licensed; the detector it wraps is AGPL-3.0.

OVERVIEW

Across 2022–2023 I shipped four versions of "detection + tracking + ReID": single-person tracking on a Loomo robot, multi-person pose tracking, a ReID demo on Colab, and the perception module of a full ROS autonomy stack. Each solved a real problem in its context. None of them shared an interface; all four ran on dependency stacks that aged out.

`perception-pipeline` is the consolidated re-do against the 2026 stack. v0.1 ships detection and tracking (modern YOLO, ByteTrack); the roadmap adds BoT-SORT / OC-SORT, OSNet-class ReID, MMPose-compatible 2D and 3D pose, and TensorRT export for Jetson-class hardware.

WHAT V0.1 SHIPS (JUNE 2026)

A runnable detector + tracker loop with the architecture the rest of the roadmap will plug into.

- **Narrow interfaces**: `Detector` and `Tracker` abstract base classes; every primitive lives behind one. Swap modules without touching the rest.
- **YOLO detection**: `ultralytics` wrapper, any checkpoint (`yolov8n` by default).
- **ByteTrack association**: via the `supervision` package; persistent track IDs across frames.
- **CLI demo**: `perception-demo --input video.mp4 --output annotated.mp4`. Latency summary printed on exit (per-module mean / p95, effective FPS).
- **Type contracts**: `Detections`, `Tracks`, `FrameResult` carry per-module latencies so the runtime can budget downstream stages.
- **MIT-licensed, pip-installable**: `pip install -e .` and you're running. The detector it wraps, `ultralytics`, is AGPL-3.0, and that licence applies to anything you ship with it.

The full case: theoalessandro.com/projects/perception-pipeline

Object Detection, Multi-Object Tracking, YOLO, ByteTrack, PyTorch

theoalessandro.com/projects/perception-pipeline
github.com/thrmnn/perception-pipeline

PRACTICE · Open source · 2026

Finding the moment a robot got lost

Mine end to end: I built the detectors, ran them on public recordings and graded every result myself

Hidden	Robots that lose track of where they are while still reporting high confidence.
Revealed	Four detectors that find those moments in the logs a fleet already records, run unchanged on recordings from five platforms.
Changed	Published with what it missed, and nearly every number recomputes from the repository with one script.



PR2, MIT Stata Center dataset (CC BY 3.0): in one corridor, sure it was in another

Method public: github.com/thrmnn/ros2-localization-triage

PROBLEM

A robot misbehaves at a customer site and someone has to comb the bag files and logs to find out why.

BUILT

Four independent fault detectors for ROS 2/ROS 1 localization pipelines (sensor dropout, transform jumps, covariance spikes, pose divergence), tested against 108 minutes of real recordings from five robot platforms (Cartographer backpack rigs, a Tiago, a MiR100 AGV, a PR2, a handheld 3D rig), all public third-party benchmark data, not a live customer fleet. Every graded figure recomputes from a committed artifact with one script, `check_numbers.py`; the cross-platform flag rates recompute from the flag count and duration stated in their own row, bag linked. Clone the repo and run it.

WHY HARD

The same frozen thresholds did not transfer. The covariance threshold, set on a simulated TurtleBot3, sits inside a real Tiago's healthy yaw noise and fired five times on a recording with nothing wrong, graded wrong in the case log. A threshold is a property of the machine it was measured on, not of the failure. That is cross-platform transfer, not fleet calibration; calibrating to a live fleet's own robots is what an engagement builds.

The full case: theoalessandro.com/projects/ros2-localization-triage

ROS 2, Log Analysis, Fault Detection, Localization, Python

theoalessandro.com/projects/ros2-localization-triage
github.com/thrmnn/ros2-localization-triage

Open source · 2026

A harness for AI agents

Hidden	Agent failures nobody sees: a tool call breaks, nothing traces it, and a user finds out first.
Revealed	A typed tool layer with one structured trace line per call, and a regression suite wired into CI.
Changed	A bad change to a tool fails the build instead of reaching the user.

OVERVIEW

Agentic products hit the same walls: flaky tool calls, failures nobody can see, no evals, no safe way to ship a prompt change. A harness around the agent makes it behave like a production system: a typed tool layer, traces you can actually read, and an eval suite that catches regressions before users do.

WHAT THIS COVERS

- Typed, versioned tool layer (testable in isolation)
- An example keyword-search tool (`kb_search`) that demonstrates the eval harness on a retrieval-style task
- One structured trace line per tool call, small enough to grep
- Regression eval suite wired into CI

Status: v0.1, deliberately minimal. A reference implementation of the pattern, plain enough to copy into your own codebase rather than depend on.

The full case: theoalessandro.com/projects/agent-harness

As committed (bdbe373)

```
$ python -m agent_harness.evals
  examples/eval_cases.py
suite: kb-example
  PASS  lidar-query-finds-lidar-doc
  PASS  hit-carries-title
  PASS  top-k-is-respected
  PASS  unknown-terms-return-nothing
  PASS  title-match-outranks-body-match
  PASS  fetch-returns-full-document
6 passed, 0 failed
exit 0
```

After one change in examples/kb_search.py: TITLE_WEIGHT = 2 → 1

```
$ python -m agent_harness.evals
  examples/eval_cases.py
suite: kb-example
  PASS  lidar-query-finds-lidar-doc
  PASS  hit-carries-title
  PASS  top-k-is-respected
  PASS  unknown-terms-return-nothing
  FAIL  title-match-outranks-body-match
        equals('hits.0.id', 'doc-evals'):
        expected 'doc-evals', got
        'doc-retrieval'
  PASS  fetch-returns-full-document
5 passed, 1 failed
exit 1
```

The repository's own eval suite at commit bdbe373, run twice: as committed, all six cases pass and the run exits 0; after one change to the example search tool (TITLE_WEIGHT from 2 to 1, as the suite's comment describes), the regression case fails and the run exits 1.

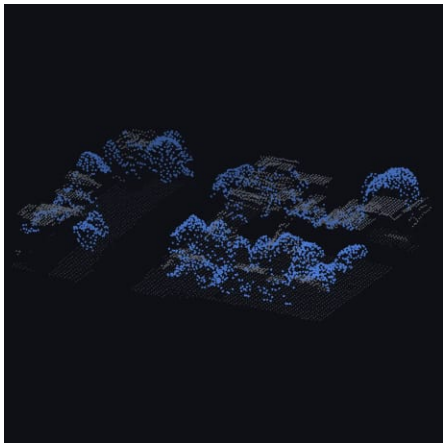
AI Agents, Typed Tool Layer, Evals, Tracing, Agentic Systems

theoalessandro.com/projects/agent-harness
github.com/thrmnn/agent-harness

MORE WORK · Senseable City Lab, MIT and AMS Institute · since 2024

Leaf area from an aerial scan

Hidden	How much leaf a city's trees carry. Registers record where each tree stands, not how much shade it gives.
Revealed	Leaf area per tree, recovered by a physical inversion from the aerial LiDAR survey the city already flies.
Changed	An estimate that can be argued with on its own terms, from data that already exists.



Jordaan, Amsterdam · AHN5, the open Dutch airborne laser scan

THE PROBLEM

Municipal tree registers are good at the things a register is for: species, location, trunk diameter, when it was last inspected. They do not record how much leaf a tree actually carries, which is the quantity that matters for shade, cooling and interception. Measuring it directly, tree by tree, across a city is not something anyone is going to fund.

WHAT I BUILD

Cities already fly aerial LiDAR surveys on a regular cycle for entirely unrelated reasons. Those scans contain the information: a pulse that passes through a canopy and a pulse that is intercepted by it are distinguishable, so the fraction of sky visible through the crown can be recovered from data that already exists.

From that gap fraction, leaf area follows by a physical inversion rather than a fitted model, which is the part I care about most. The relationship between how much light a canopy blocks and how much leaf it carries is classical physics with known assumptions, so the estimate can be argued about on its own terms instead of defended as the output of a regressor nobody can inspect. An earlier version of this work did use machine learning; the current method does not.

WHAT IS SHOWN HERE

Method only: there are no numbers, figures or findings on this page.

Remote sensing, LiDAR, Canopy, Beer-Lambert inversion, Python

theoalessandro.com/projects/aerial-lidar-tree-census