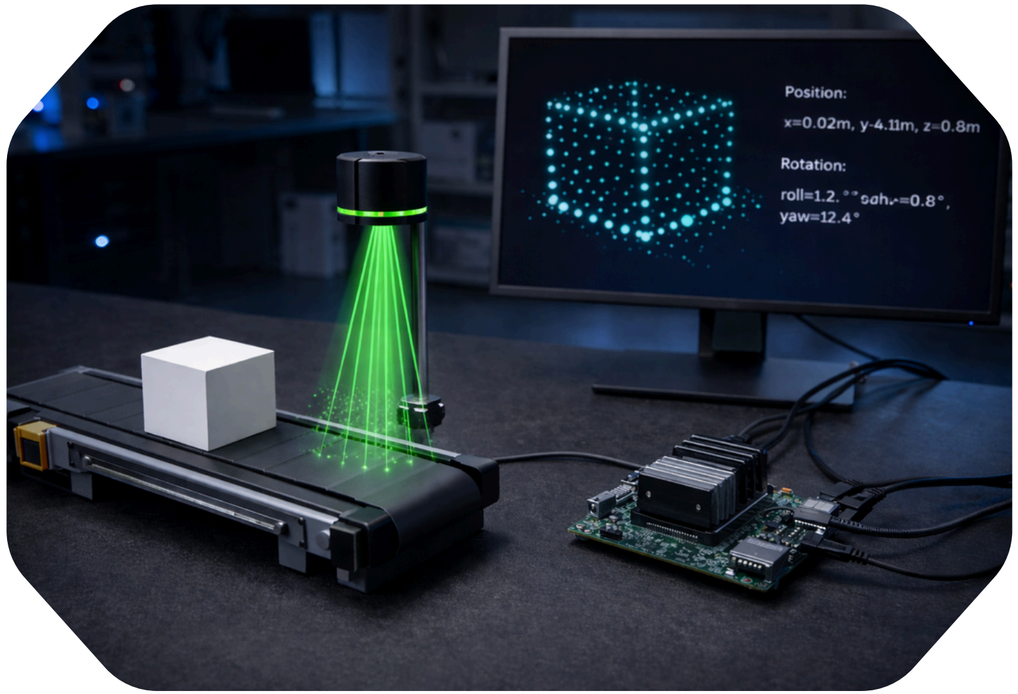
Study of a 3D Object Position and Orientation Detection System with Embedded AI Processing



Supervisor : Erwan Dupont

Porteurs : Laurent Petit et Hani Al Hajjar

- Andri Halili
- Syed Muhammad Abubakar
- Houssem Abbes
- WangBiyao
- Youssef el gouri

Project Context & System Setup

Goal: *Develop a 3D object position and orientation detection system using:*

- Laser slicing technique (2D) combined with a moving conveyor
- → Each laser slice corresponds to one layer, enabling 3D reconstruction
- Laser + camera fusion for measurement refinement and accuracy

The AI must run on a resource-limited embedded board.

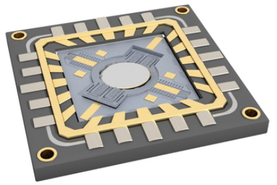
Team Structure

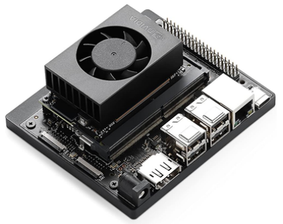
Mechatronics / Hardware (3)

- Integrate & calibrate the physical system.
- Design the optical alignment and triangulation geometry.
- Assemble the full physical prototype.

AI / Software (2)

- Generate synthetic point-cloud datasets with realistic noise.
- Develop and train a lightweight AI model.
- Convert and deploy the model on the Jetson Orin Nano.
- Optimize inference speed, memory and robustness.



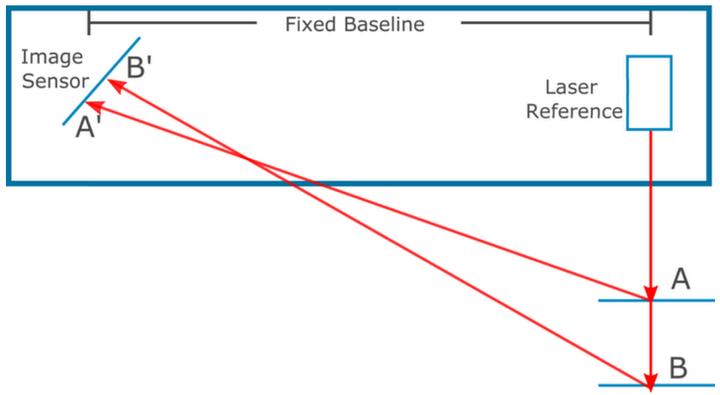

New Methodology

Initial Concept

- **Laser triangulation** using a laser diode + MEMS scanning mirror + camera
- **Goal:** full 3D surface reconstruction via controlled laser projection

Issue Encountered

- MEMS mirror malfunctioned and became unusable
- Mechanical failure occurred late in the project timeline
- Line laser is hard to use without precise angle knowledge

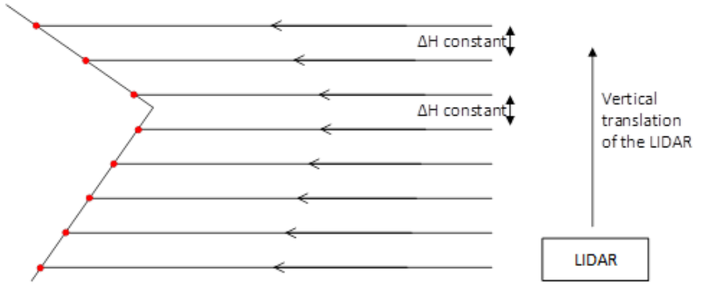


Revised Architecture

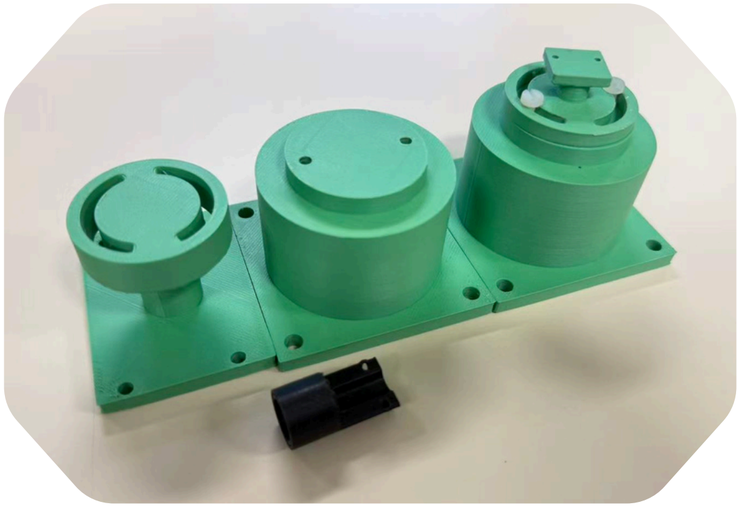
- Static vertically oriented LiDAR sensor
- Object displacement via motorized conveyor
- Mechanical motion replaces optical scanning

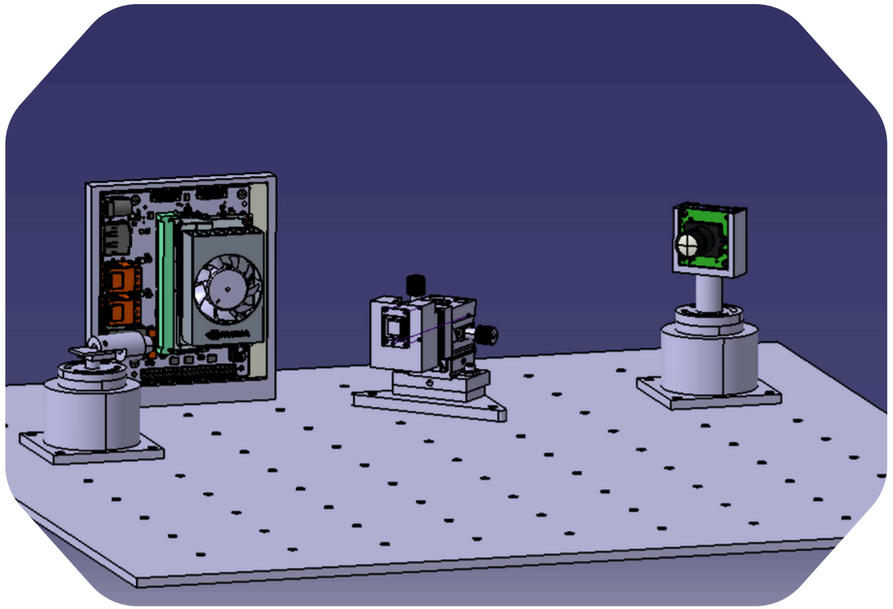
3D Reconstruction Pipeline

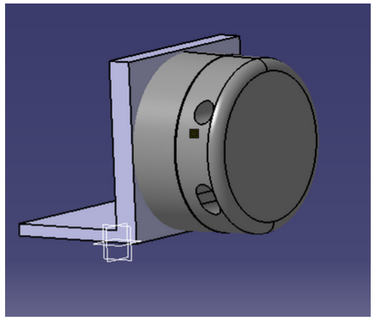
- Each LiDAR scan produces a 2D vertical profile
- Conveyor motion introduces a known translation (Δx = 1 cm)
- ROS 2 pipeline accumulates profiles into a 3D point cloud

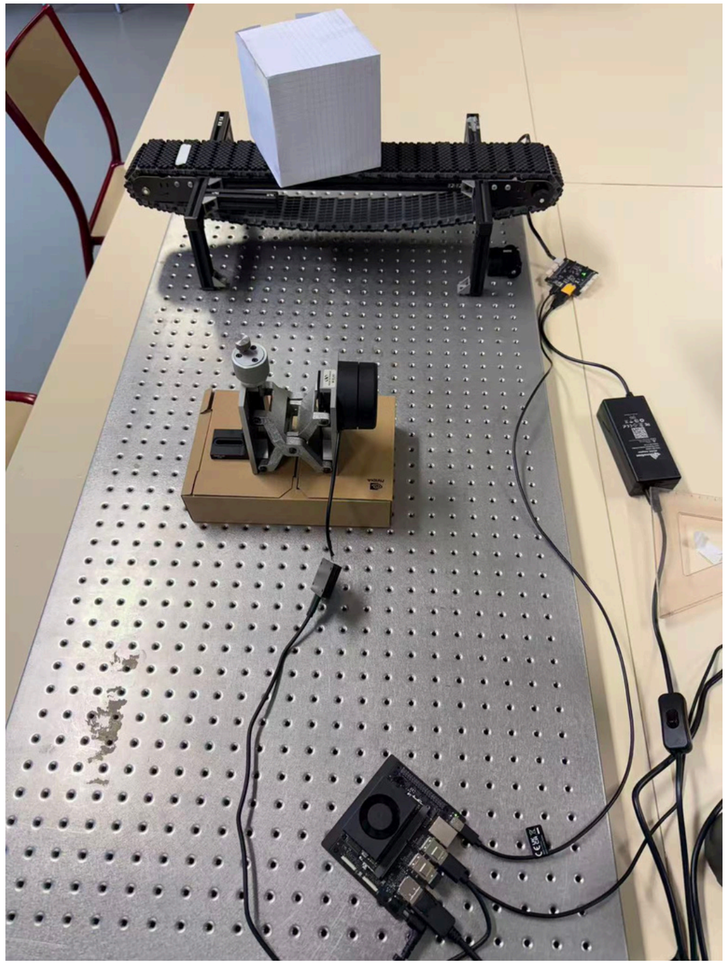


Mechanical Design & 3D printing







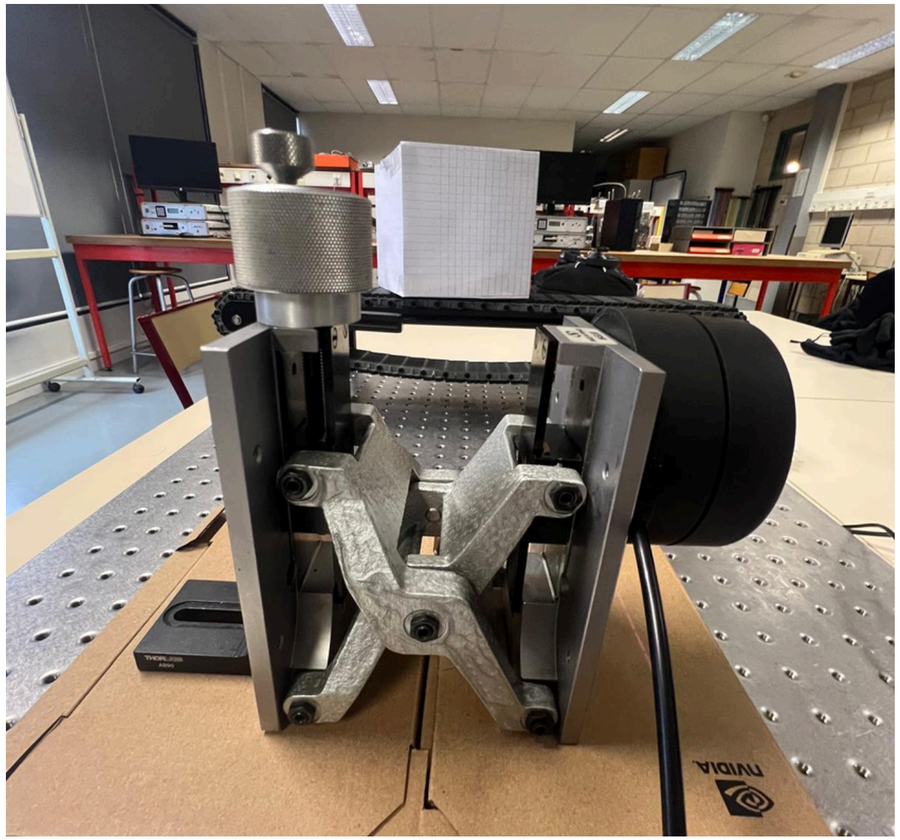


Module Assembly

3D printing for each of the components.

The mechanical architecture of the system is designed to provide a stable, modular, and reconfigurable physical platform for LiDAR-based 3D scanning and embedded pose estimation.

New Lidar System



Objective: Generate 3D Poincloud using lidar system

Goal: Use a configuration to be able to scan the object accurately enought so the software team can proceed to work on the data.
Requirement: Generate high-fidelity 3D datasets that structurally mimic the output of the final triangulation system.

System Architecture: Translational Scanning

Integration of a 2D LIDAR sensor on a vertical linear translation stage.
The system captures sequential 2D slices at controlled height levels.
Precise correlation between the linear actuator position and LIDAR profiles.

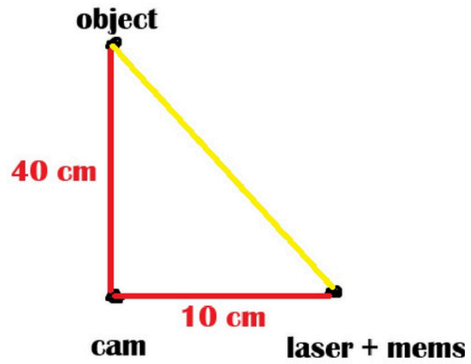
Output: Volumetric Point Clouds

Data Structure: Stacks individual 2D laser profiles to reconstruct a dense 3D Point Cloud.
Relevance: This method produces geometric data (X, Y, Z coordinates) analogous to the Laser Triangulation method, ensuring compatibility with the AI models being developed.

Triangulation : Laser point / line and the camera

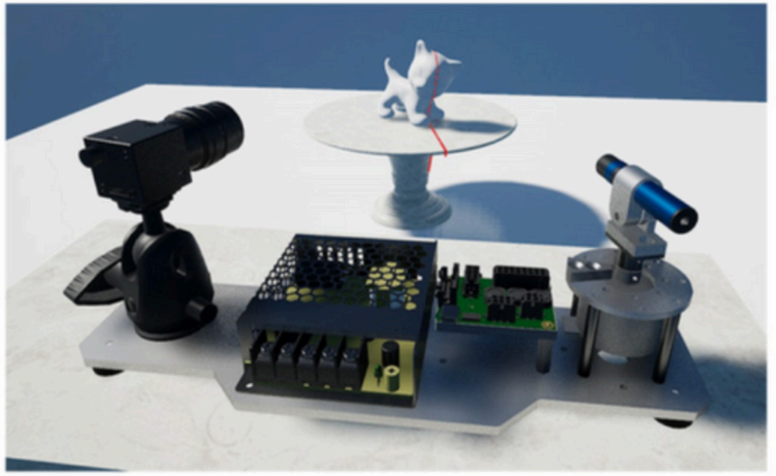
Initial Approach

- Calibrate the camera
- Install the setup in the forgiving geometry
- Use the camera filter to detect the red dot of the laser
- Using the equations that we showed last time, built in an algorithm, we get the coordinates of the point.



Laser Line System Challenges

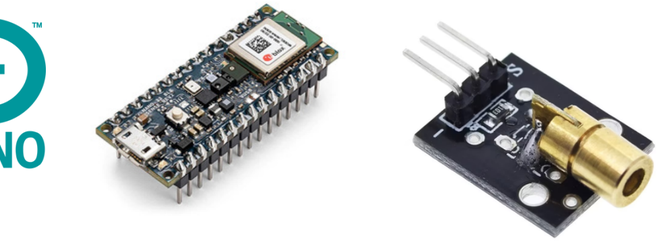
- Unknown projection angle → wrong coordinates
- Requires laser spanning full camera height
- Calibration needs constant reference objects
- Conclusion: Not practical for this setup



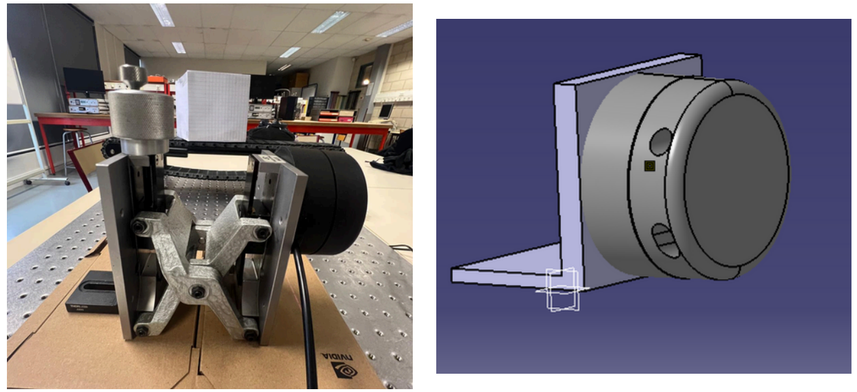
(a)

Calibration of laser and Lidar setup

- Calibrated by interfacing with Arduino Nano 33 BLE Sense board
- The validation step verified electronic control of the laser
- Ensured beam stability
- Guaranteed reliable and precise light source for the measurement system
- The use of optical element to transform the laser dot to a laser line. The problem is that the intensity of the beam is too low, which makes the line really small and we weren't able to work with. The application of the laser line was built than with a red line paper pretaped on top of the object.



- **Initial approach:** Manual translation stage (Newport M-LNR type) for LiDAR positioning
- **Strategic shift:** Moved toward system automation to optimize experimental process and ensure better repeatability
- **Objective:** Fix LiDAR at 15-20 cm height
- **Final solution:** Programmable control system for precise and autonomous part movement via software interface



Hardware Validation – Linear Positioning System

Technical Validation of the Newport CONEX-CC System

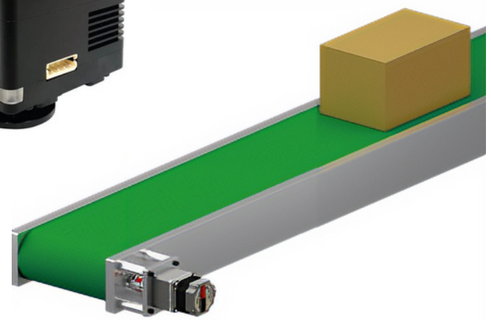
- **Equipment:** Controller + MFA-CC linear stage (ultra-compact DC servo).
- **Commissioning:** Completed with the GUI interface installed and USB communication validated.
- **Verified Performance:** Smooth and stable motion thanks to the low-noise amplifier. Precision specifications confirmed: minimum step size of 100 nm with repeatability of ±2.5 µm.
- **Conclusion:** System conforms to manufacturer specifications

Identified Limitation

- **Constraint:** Maximum travel range limited to 25 mm.
- **Impact:** Insufficient to cover the measurement zone required by the LiDAR sensor.
- **Risk:** Lack of flexibility for the experimental scenario.

Conveyor Servo Motor

- LSS Config software used to configure and test the conveyor servomotor
- USB communication successfully established
- Servomotor automatically detected and configured
- Manual control tests validated smooth and stable motion
- System confirmed ready for experimental integration



ROS2 Architecture for Lidar Scanner

Step 1: Angle limitation

We will use an algorithm to keep only the information between -15° and 15°.
Purpose: Scan only the object to limit the noise and the unnecessary scan of the walls and other parts.

Step 2: Filtering the noise

- Range filtering, to remove points outside the region of interest.
- Outlier removal, to suppress isolated or inconsistent measurements,

This is basically done using MORPH_OPEN and MORPH_CLOSE.

Step 3: 2D TO 3D

Save each slice of the lidar on press and convert it to 3D point cloud. Each study of object is composed of 15 slices (15 cm) and proceeds to the next step automatically.

Step 4: Point cloud to CSV

Afterwards, we will transfer this pointcloud to a CSV that allows the Software team to use the data to simulate and work on the object easier.

Synthetic Data Generation

The Problem

- Only 2 real scans captured due to hardware constraints.
- Training a 6DOF model requires thousands of labeled samples.

The Solution

- Raycasting-based LiDAR simulator.
- Generate 15,000 samples in minutes with perfect ground truth labels.

Dataset Statistics

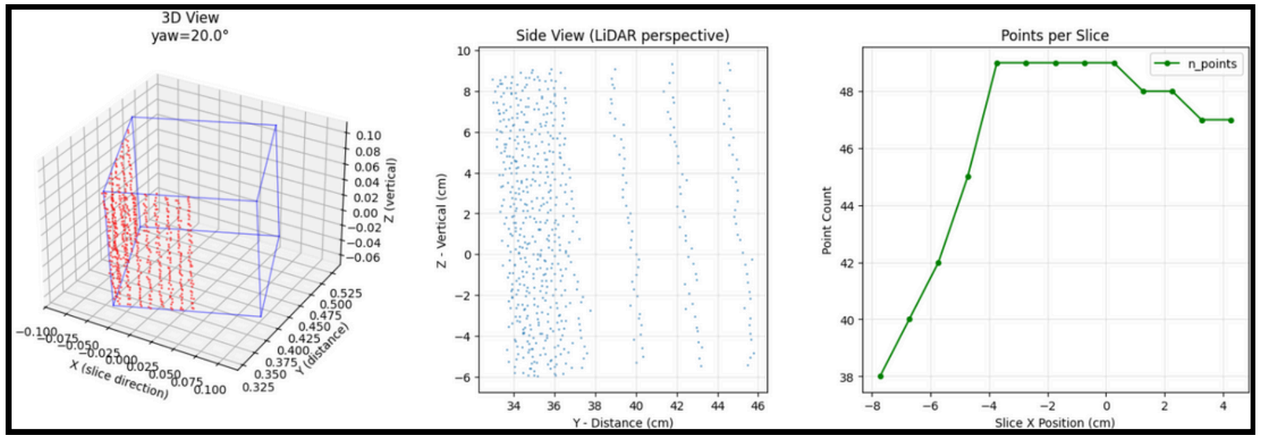
Augmentation techniques:

- **Translation:** ±5 cm random shift
- **Rotation:** Yaw ±25°, Roll/Pitch ±5°
- **Gaussian Noise:** σ = 1–5 mm
- **Dropout:** Random point removal

Simulator Parameters:

- **Angular Resolution:** 0.5°
- **Field of View:** 90°
- **Range Noise σ:** 2 mm
- **Angular Noise σ:** 0.1°

Metric	Value
Total Samples	15,000
Points per Sample	200–600
Slices per Sample	10–15
Train/Test Split	80% / 20%



Feature Selection

Problem

Target: Jetson Orin Nano (real-time constraint)
Required latency: <100 ms (target <10 ms)
End-to-end point-cloud models (PointNet/++): too large and expensive

The Solution

- Hand-crafted statistical features reduce input to fixed 150 dimensions
- Enables a lightweight MLP (~17K parameters, <100KB)
- Encodes domain knowledge about LiDAR scanning geometry
- Result: >1000 FPS inference — 100× faster than real-time requirement

Advantage	Explanation
1. Fixed Dimensionality	Always 150 inputs regardless of point count. No padding, masking, or dynamic sizing needed.
2. Noise Robustness	Statistics (mean, std) naturally smooth out individual point noise and outliers.
3. Computational Efficiency	Simple MLP is sufficient. No costly graph convolutions, attention, or symmetric pooling operations.
4. Data Efficiency	Domain knowledge is baked in. Requires far less training data than end-to-end learning (~15K samples vs. millions).
5. Embedded-Friendly	Small model fits in cache, minimal memory bandwidth, ideal for edge deployment.

```
============================================================
DATASET STATISTICS
============================================================

Pose distributions:
  x: mean=-0.01cm, std=1.16cm, range=[-2.00, 2.00]cm
  y: mean=42.96cm, std=2.91cm, range=[38.00, 48.00]cm
  z: mean=2.52cm, std=1.45cm, range=[0.00, 5.00]cm
  roll: mean=0.01°, std=2.87°, range=[-5.00, 5.00]°
  pitch: mean=0.02°, std=2.90°, range=[-5.00, 5.00]°
  yaw: mean=0.04°, std=14.36°, range=[-25.00, 25.00]°

Feature distributions (non-empty slices):
  z: mean=-0.0219, std=0.0398, range=[-0.1151, 0.0743]
  n_points: mean=46.9964, std=4.7054, range=[1.0000, 59.0000]
  x_min: mean=-0.0479, std=0.0152, range=[-0.0856, 0.0197]
  x_max: mean=0.0982, std=0.0152, range=[0.0000, 0.1343]
  x_mean: mean=0.0252, std=0.0149, range=[-0.0156, 0.0671]
  x_std: mean=0.0429, std=0.0011, range=[0.0000, 0.0460]
  y_min: mean=0.3537, std=0.0354, range=[0.0000, 0.4998]
  y_max: mean=0.3681, std=0.0382, range=[0.0000, 0.5551]
  y_mean: mean=0.3607, std=0.0364, range=[0.0000, 0.5138]
  y_std: mean=0.0035, std=0.0030, range=[0.0000, 0.0509]

Points per sample: mean=611, range=[515, 716]
Slices per sample: mean=13.0, range=[13, 14]
Saved train_X.csv and train_Y.csv
```

AI Model Architecture

Huber Loss Function

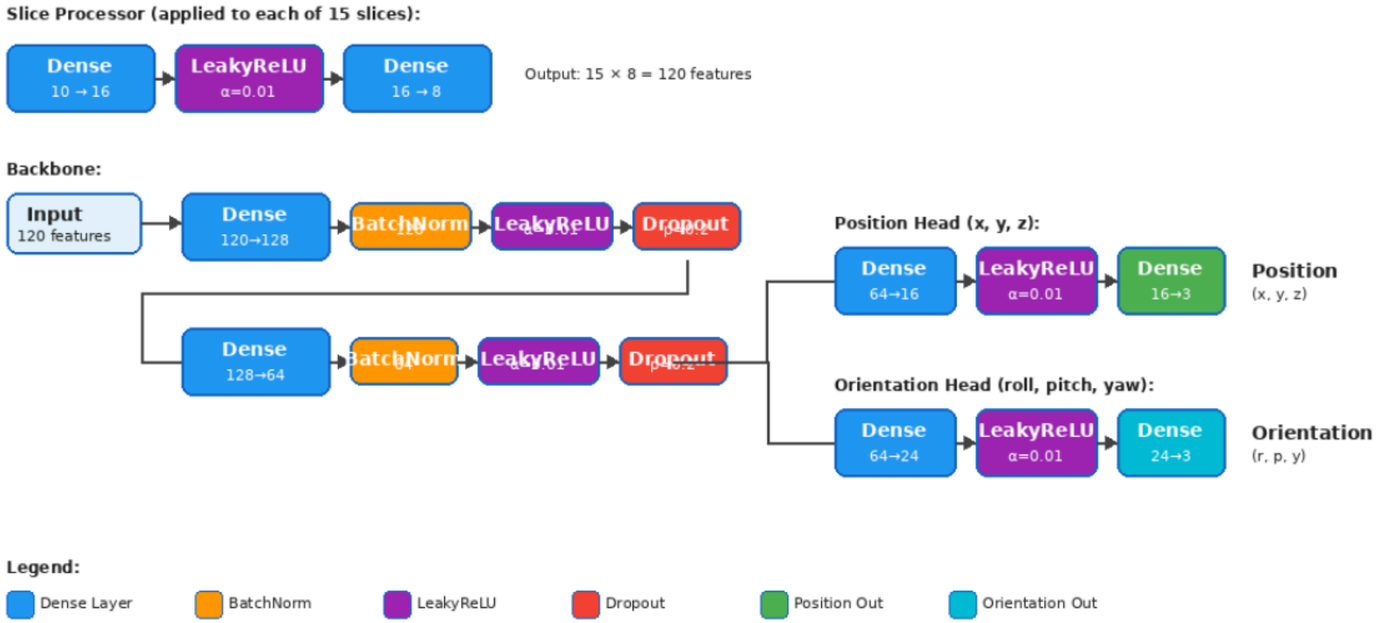
$$L = w_{pos} \times H(pos) + w_{ang} \times H(roll,pitch) + w_{ang} \times w_{yaw} \times H(yaw)$$

Why Huber? **Robust to outliers** — behaves as MSE for small errors, linear for large errors.

Features

Slice-wise Processing: Each slice processed identically with shared weights. Respects spatial structure.
Dual Output Heads: Orientation head larger (24 neurons) than position (16) — angular estimation is harder, needs more capacity.
Regularization: Batch Normalization + 20% Dropout + Leaky ReLU. Prevents overfitting on synthetic data.

Parameter	Value
Framework	PyTorch
Hardware	Google Colab T4 GPU
Optimizer	AdamW
Learning Rate	0.001
Batch Size	64
Max Epochs	200
Early Stopping	25 epochs patience



Deployment on Jetson

Embedded Platform Setup

- Reflashed NVIDIA Jetson Orin Nano with ROS2-compatible Ubuntu
- Installed and configured ROS2 Humble distribution

Hardware Integration

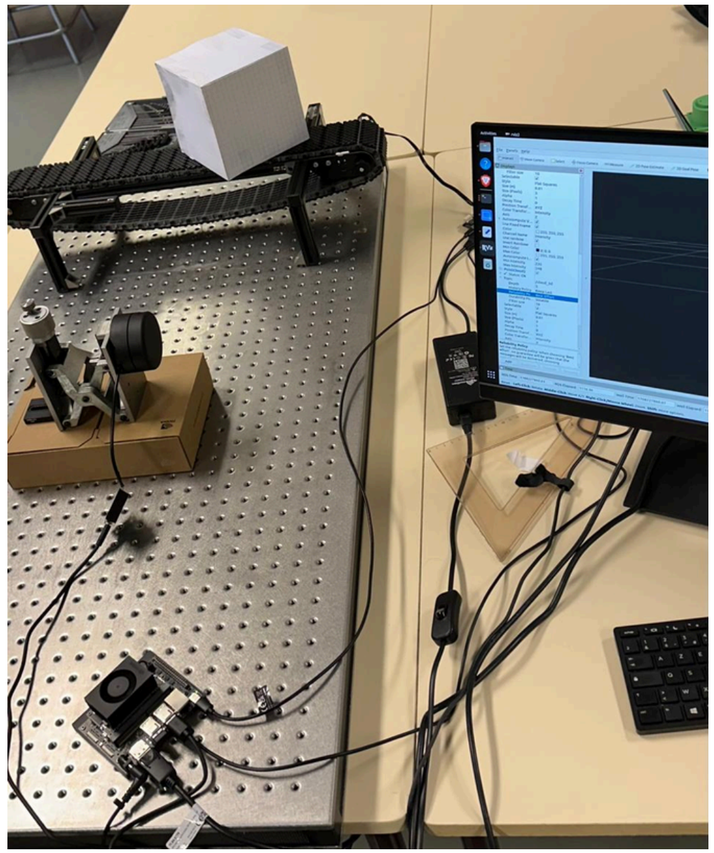
- Fixed ROS2 application for Jetson (LiDAR scanning, filtering)
- Established LiDAR serial communication
- Integrated conveyor control via Python library

Visualization & AI Deployment

- Established RViz2 for real-time point cloud visualization
- Exported and deployed ONNX model package
- Connected LiDAR output directly to AI input pipeline

Full Automation

- Created a unified launch application for the complete system startup
- Automated entire workflow from scanning to pose estimation

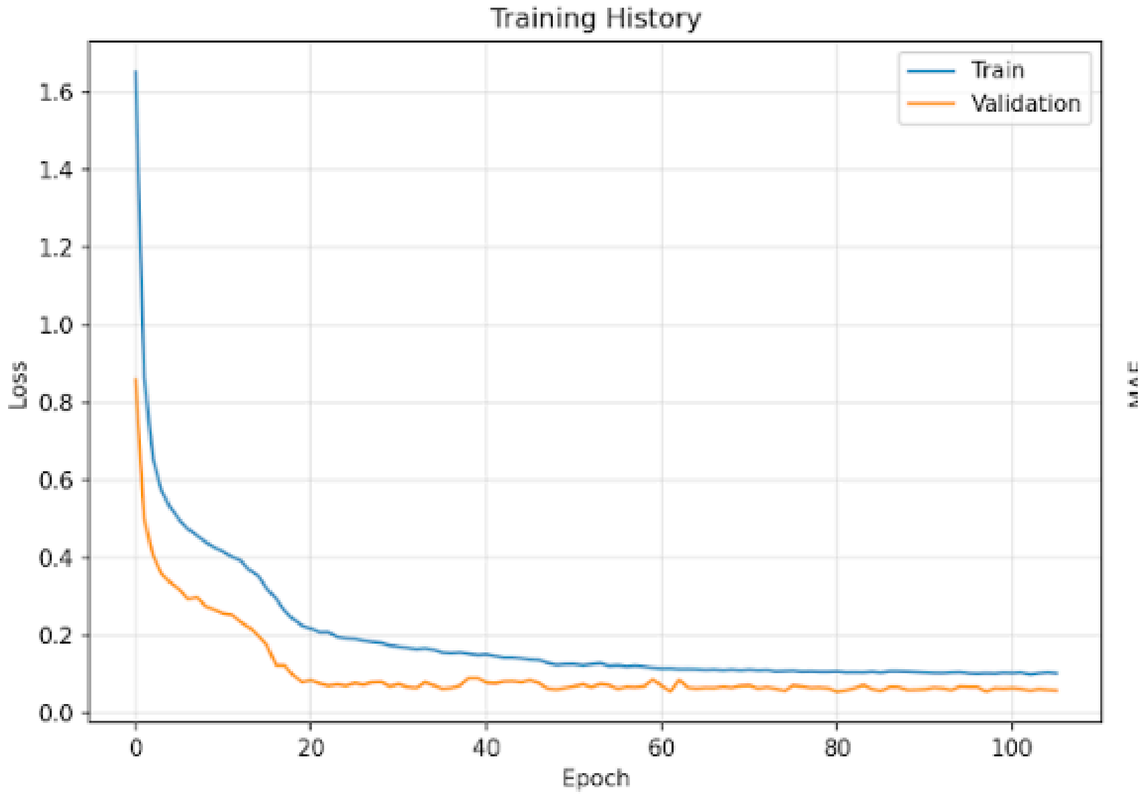


Successfully tested complete pipeline

ROS2 → Scan → Data acquisition → AI inference → Pose output

Results Obtained

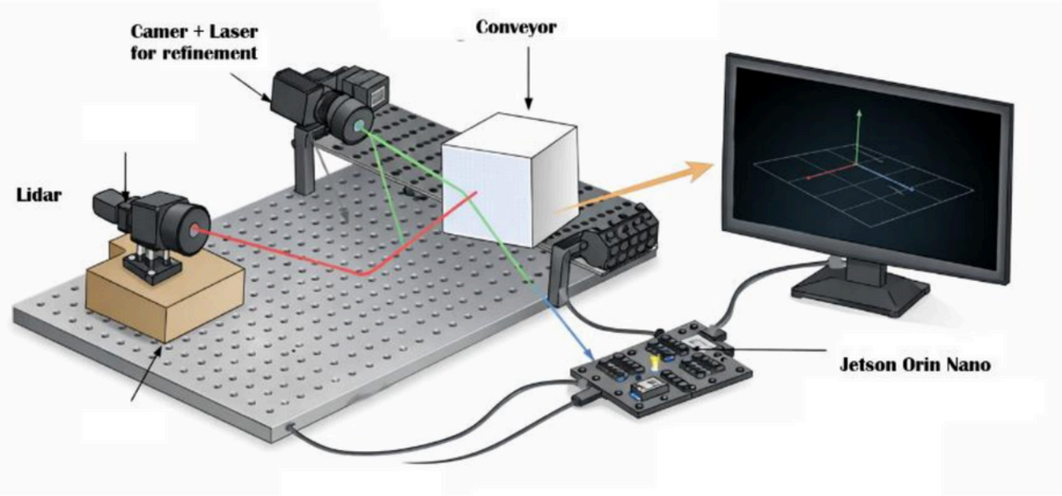
Training Results

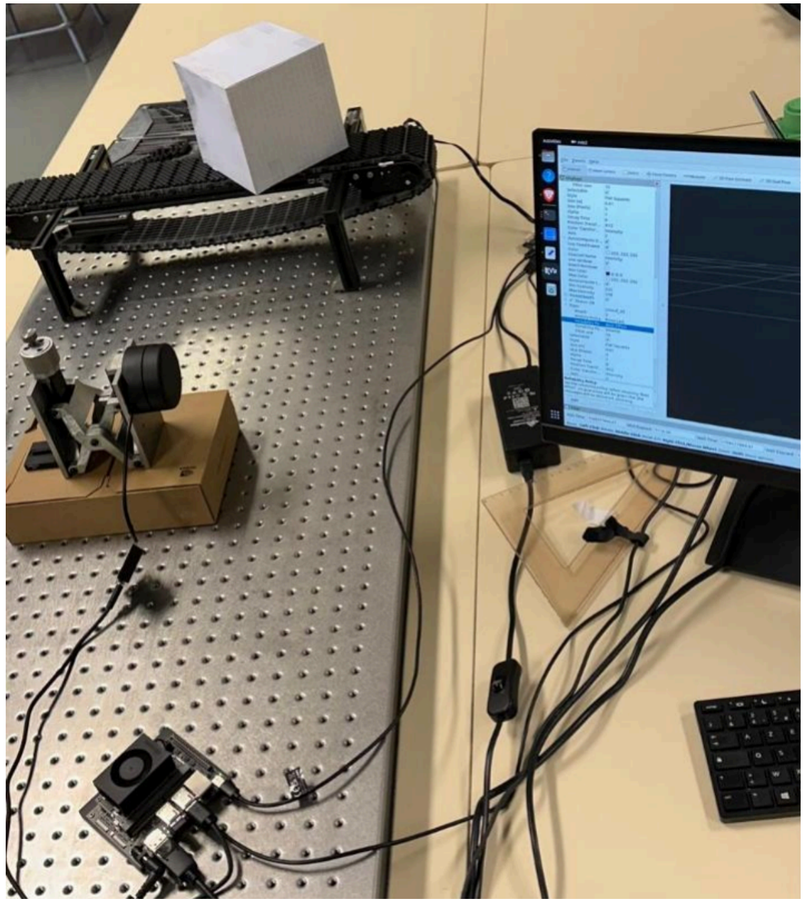


Test Results

```
============================================================
RESULTS SUMMARY
============================================================

📊 K-FOLD CROSS-VALIDATION:
----------------------------------------
  Position MAE: 2.26 ± 0.13 mm
  Angular MAE: 0.58 ± 0.03 °
  Yaw MAE: 1.04 ± 0.13 °

🎯 FINAL TEST SET RESULTS:
----------------------------------------

  Position Errors:
    X:   1.48 mm
    Y:   2.65 mm
    Z:   1.86 mm
    Avg: 2.00 mm

  Angular Errors:
    Roll:  0.28°
    Pitch: 0.37°
    Yaw:   1.37°
    Avg:   0.68°
```

Final Product





Conclusions & Future Works

✅ Achievements

- Functional 6DOF pose estimation system with LiDAR + embedded AI
- Real-time performance: <1ms inference on Jetson Orin Nano
- Successfully pivoted from broken MEMS → vertical LiDAR scanning
- Synthetic data generation overcame limited real data

🚀 Future Work:

- Integrate Camera + Laser for refinement
- Multiple object types
- Full 360° rotation support

💡 Key Innovations

- Vertical slice-stacking: 2D LiDAR → 3D reconstruction
- Feature-based approach: 150D vectors, 17K parameters
- Symmetry-aware training for cubic objects

🌍 Applications:

- Quality control & inspection
- Robotic manipulation
- Autonomous systems

Thank You