MANTIS-4D: End-to-End Autonomous Robotic Injection with Multi-View Pose Estimation and Gaussian Splatting

Konsta Kiirikki
School of Science and Technology
Aalto University
konsta.kiirikki@aalto.fi

March 1, 2026

Abstract

We present MANTIS-4D, an end-to-end autonomous robotic system designed for supervised injection-style manipulation tasks. The system combines a mobile omnidirectional platform with a table-mounted robotic hand and a multi-camera perception stack. To localize in previously unseen rooms, MANTIS-4D builds a scene representation on-the-fly using Gaussian splatting. Wrist pose estimation is performed by a novel *Multi-View Temporal PoseNet* (MVTP-Net) that fuses bird's-eye and wrist-camera streams through a shared Vision Transformer backbone, cross-attention spatial fusion, and a temporal self-attention module over the action history. Target detection uses a fine-tuned YOLO11n detector that localizes manipulation targets from the bird's-eye view. A tangent-constrained trajectory planner converts estimated poses into safe, smooth motion primitives enforcing strict velocity and acceleration limits. We describe the architecture, training pipeline, calibration procedure, and safety mechanisms of the integrated system.

1 Introduction

Deploying robots in unstructured environments for manipulation tasks that require millimetre-level precision—such as medical injection procedures— remains an open challenge. Three tightly coupled problems must be solved simultaneously: *(i)* scene-level localization and mapping to position the platform relative to the workspace, *(ii)* fine-grained perception of the manipulator and the target, and *(iii)* safe, constrained motion execution with real-time operator oversight.

Prior work has addressed these problems largely in isolation. Neural radiance fields and Gaussian splatting [1] have demonstrated high-quality scene reconstruction but are rarely integrated into closed-loop robot control. Multi-view pose estimation networks have been proposed for human body pose and object tracking, but their application to end-effector state estimation in imitation-learning settings remains limited. Safety-critical manipulation planners typically rely on analytical models that are difficult to adapt to objects with variable geometry.

MANTIS-4D addresses these limitations in an integrated system built for a concrete demo task: given a previously unseen room, the platform must autonomously map the environment, navigate to a known operating table, estimate the wrist pose of a table-mounted so-101 arm from multi-view video, localise a manipulation target (can/vial proxy), and execute a tangent-aligned insertion trajectory—all under continuous operator supervision with a hard E-stop. Our main contributions are:

- A **Multi-View Temporal PoseNet** (MVTP-Net) that estimates end-effector pose from synchronised top-view and wrist-view RGB streams, using cross-attention fusion and a temporal transformer.
- A **Gaussian-splat-based localization** pipeline for real-time map building and platform pose estimation in novel environments.

Figure 1: Rendered 3d scene of the recording.

- A **fine-tuned YOLO11n** detector for bird's-eye-view target localisation, trained on a custom tabletop dataset.
- A **tangent-constrained trajectory planner** that produces safe, smooth injection-style motion primitives from estimated poses.
- An end-to-end **system integration** including multi-camera calibration, a REST-based backend, a real-time operator UI, and hardware-level safety interlocks.

2 Related Work

Radiance-field and Gaussian-splat scene representations for robotics. Neural implicit scene representations have enabled photo-consistent mapping and analysis-by-synthesis localization, but their optimization cost and rendering throughput can be a bottleneck for closed-loop robotics. NeRF-SLAM couples dense monocular SLAM with hierarchical neural radiance fields to reconstruct scenes in real time while leveraging SLAM-derived pose and depth supervision [2]. A major shift came with 3D Gaussian Splatting (3DGS), which represents scenes as explicit 3D Gaussian primitives and achieves high-quality real-time novel-view synthesis at 1080p [1]. This explicit, differentiably-renderable representation has rapidly entered SLAM and navigation pipelines: SplaTAM uses splatted Gaussian maps with differentiable rendering to jointly optimize camera poses and the map in dense RGB-D SLAM [3], while Gaussian Splatting SLAM demonstrates a monocular 3DGS-based SLAM system that relaxes the requirement of offline SfM poses [4]. Beyond mapping, Splat-Nav shows that GSplat maps can support safe real-time robot navigation by combining vision-based pose estimation from RGB with safe corridor planning [5]. In contrast to works that treat mapping and localization as the end objective, MANTIS-4D uses on-the-fly Gaussian-splat mapping as a deployment-time localization substrate and operator-facing map that enables downstream precision manipulation in previously unseen rooms.

Multi-view and temporal 6-DOF pose estimation. Modern 6D pose estimation spans render-and-compare refiners and foundation models, often relying on object-centric assumptions and sometimes depth sensing. MegaPose estimates the 6D pose of novel objects given a region-of-interest and a CAD model via a render-and-compare pipeline trained on large-scale synthetic data [6]. FoundationPose unifies pose estimation and tracking for novel objects in both model-based and model-free setups using RGB-D input and transformer-based architectures [7]. Multi-view consistency can substantially reduce single-view ambiguities; CosyPose combines information from multiple RGB views to jointly recover object and camera poses via global refinement [8]. For temporal stability in videos, transformer models that attend over previous frames have been explored for 6D pose refinement in video sequences [9]. For rotation regression we adopt the continuous 6D rotation representation of Zhou et al. [10]. MANTIS-4D differs from object-centric pose pipelines by focusing on end-effector (wrist) state estimation: MVTP-Net fuses synchronized wrist-view and

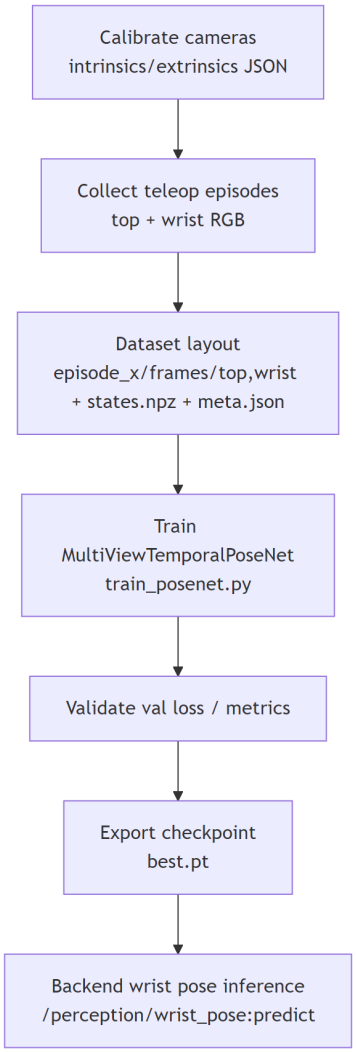

Figure 2: Overview of the MVTP-Net wrist-pose estimation pipeline. Synchronised top-view and wrist-view frames are encoded by a shared ViT backbone; a cross-attention block fuses the two views; a temporal transformer processes the action sequence; and linear heads output Δposition and 6D rotation deltas.

bird's-eye-view streams through cross-attention and temporal self-attention to produce smooth pose deltas suitable for control.

Teleoperation, imitation learning, and visuomotor policies. Imitation learning is a pragmatic route to contact-rich manipulation but faces distribution shift in sequential prediction; DAgger formalizes this regime via a no-regret reduction that motivates data aggregation under the learner's induced state distribution [11]. Recent approaches scale imitation by modelling action sequences with modern generative models: ACT learns a generative model over action chunks with transformers (trained as a conditional VAE) paired with a low-cost teleoperation system [12], while Diffusion Policy represents visuomotor control as conditional denoising diffusion over action sequences [13]. For mobile manipulation, Mobile ALOHA extends teleoperation and behavior cloning to whole-body tasks by augmenting a manipulation platform with a mobile base [14]. In MANTIS-4D, teleoperation demonstrations are used primarily to supervise perception (multi-view temporal wrist pose estimation), while motion execution remains an explicitly constrained, confidence-gated primitive with operator-visible decisions and hardware safety interlocks.

Real-time object detection for manipulation. Real-time object detection has been dominated by single-stage architectures such as YOLO, which reframes detection as direct regression from pixels to bounding boxes enabling

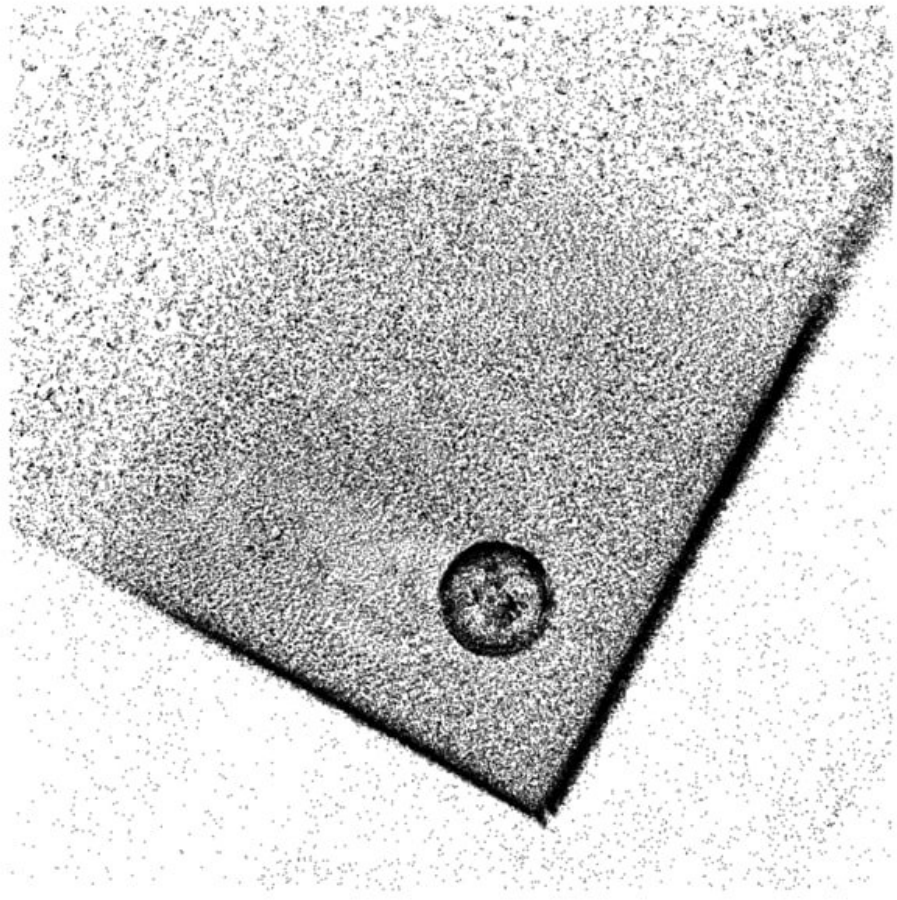

Figure 3: The collapsed 2d pointcloud creates limits for the robot to explore.

end-to-end optimization and real-time throughput [15]. We adopt a lightweight YOLO11n model for bird's-eye target localisation [16], using only the normalised target centroid and size proxy as a perception primitive that feeds a tangent-constrained insertion planner with explicit velocity and acceleration limits and abort conditions.

3 System Overview

MANTIS-4D consists of three hardware units and a backend compute stack (Figure 4):

1. **Mobile platform (Kiwi omni)** carrying a forward-facing USB webcam used for mapping and navigation.
2. **Table-mounted robotic hand (so-101, 12 V)** equipped with a USB webcam mounted tightly to the wrist, pointing forward.
3. **Bird's-eye tripod camera** — a standard USB webcam rigidly mounted on a tripod overlooking the operating table, used for target detection and top-down supervision.

All three cameras are generic USB webcams operating at 640×480 pixels and 30 fps. The backend runs on a single NVIDIA H200 SXM (141 GB VRAM) server (44-core CPU, 182 GB RAM). A containerised backend server receives sensor streams, runs inference, logs runs, and exposes a REST API consumed by a web-based operator interface. An E-stop signal propagates from the UI through the backend to the robot controller within < 300 ms. Code and trained weights are available at `https://github.com/PlayerPlanet/MANTIS-4D`; teleoperation datasets are released at `https://huggingface.co/datasets/k-kiirikki/top_wrist_so101_with_depth`.

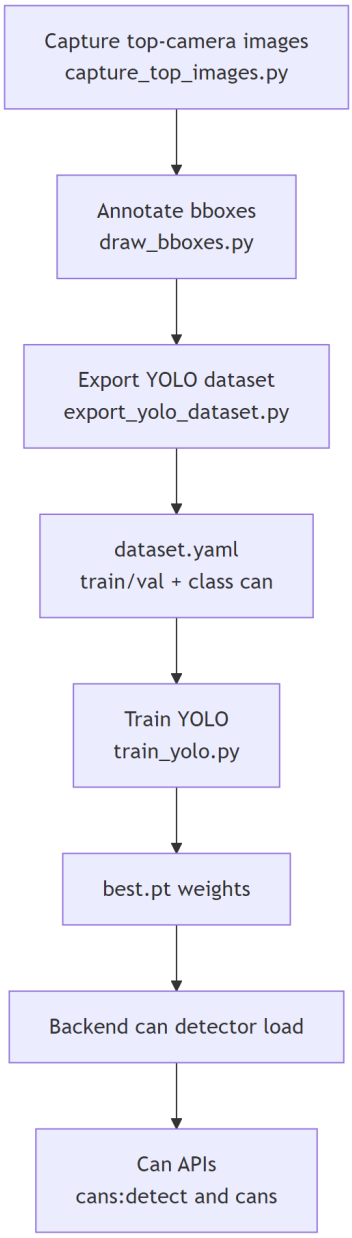

Figure 4: YOLO-based can/target detection pipeline from top-camera frames.

4 Methods

4.1 Multi-Camera Calibration

All geometric reasoning is performed in a shared *table frame*. We calibrate intrinsics and extrinsics of the three cameras using an ArUco marker board (A4 paper, ID 0, dictionary DICT_4X4_50) placed on the table.

Camera intrinsics (f_x, f_y, c_x, c_y) are estimated using OpenCV's `calibrateCamera` with a standard checkerboard. Extrinsic transforms between the bird's-eye camera, the hand camera, and the hand base frame are solved by co-observing the ArUco board from all viewpoints. The resulting calibration bundle—hand camera intrinsics, bird's-eye intrinsics, and three rigid transforms—is serialised to JSON and version-tagged with a SHA-256 digest of its parameters. Each logged run records the calibration version ID, enabling offline reproducibility.

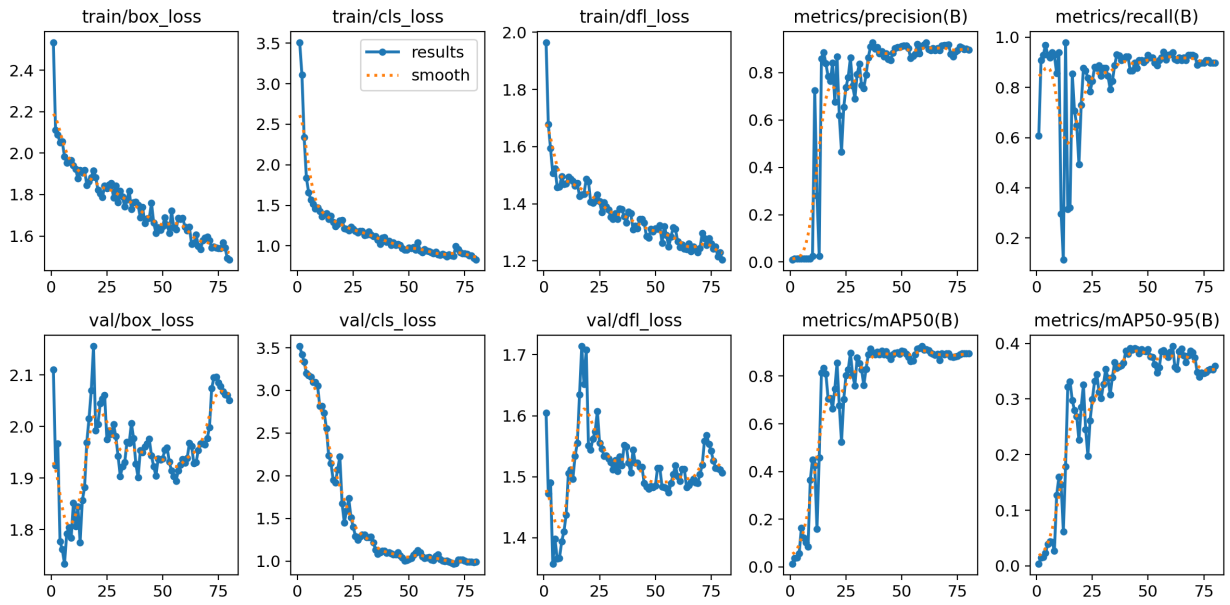

Figure 5: YOLO11n training curves. Box loss, classification loss, precision, recall, and mAP@50 / mAP@50:95 over 80 epochs on the top-camera can detection dataset.

4.2 Scene Mapping with Gaussian Splatting

During an initial exploration phase, the platform moves through the room and records an RGB video stream. Camera poses are estimated offline using COLMAP [17] structure-from-motion on the captured frames. We then train a 3D Gaussian splat using the `splatfacto` method implemented in nerfstudio [18] with default hyperparameters. The resulting splat is rendered as a top-down floorplan displayed in the operator UI, providing map coverage and platform pose feedback. Localisation within the map is maintained using odometry fused with the rendered pose; when localisation confidence drops below a threshold of 0.70, the system halts and reports a `LocalizationDegradedError`.

4.3 Target Detection with YOLO11n

We detect manipulation targets (aluminium cans used as vial proxies) in the bird's-eye camera stream using a YOLO11n model fine-tuned on a custom top-camera dataset.

Dataset. Images were captured with a fixed top camera under varying lighting conditions and object placements. Bounding boxes were manually annotated using a custom drawing utility and exported in YOLO format (single class: `can`). The dataset is split into training and validation subsets.

Training. We fine-tune YOLO11n from ImageNet-pretrained weights for 80 epochs with a batch size of 16 and input resolution 640×640. We use the Adam optimizer with cosine-annealing learning rate schedule, initial learning rate 10^{-2}, and weight decay 5×10^{-4}. Automatic mixed-precision (AMP) training is enabled.

Results. After 80 epochs the model achieves mAP@50 of **89.6%** and mAP@50:95 of **35.9%** on the held-out validation set, with precision **89.7%** and recall **89.8%** (Figure 5). Detections are post-processed to normalised bounding-box centroids $(c_x^n, c_y^n) \in [0,1]^2$ and radius, which are forwarded to the injection-spot estimator.

4.4 Multi-View Temporal Pose Estimation (MVTP-Net)

Wrist pose estimation is the core perception component. Given a temporal sequence of synchronised top-view and wrist-view RGB frames, MVTP-Net predicts per-step 6-DOF pose deltas $(\Delta\mathbf{p}, \Delta R)$.

Architecture. The network is illustrated in Figure 2 and proceeds in four stages:

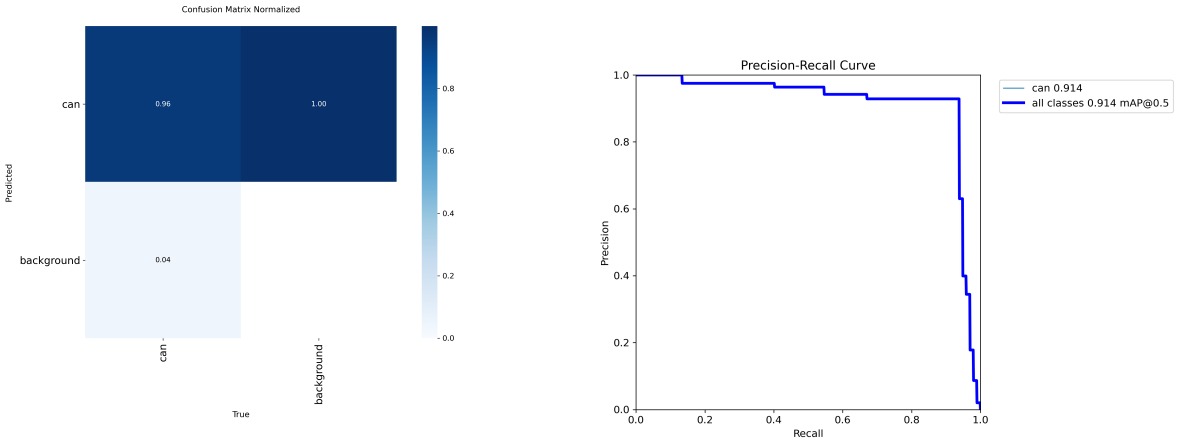

Figure 6: *Left:* Normalised confusion matrix for the can detector. *Right:* Precision–Recall curve at IoU 0.5.

1. **Shared visual backbone.** A Vision Transformer (ViT-B/16 [19], pre-trained on ImageNet-21k) serves as a shared encoder for both camera views. For a batch of B sequences of length T, all $2BT$ frames are encoded in parallel: the backbone produces a sequence of N patch tokens per frame, which are projected to a common dimension $D = 256$ via a linear layer.
2. **Cross-attention spatial fusion.** A learnable per-frame query token $\mathbf{q} \in \mathbb{R}^D$ first cross-attends to the *wrist-view* tokens and then to the *top-view* tokens via two consecutive `CrossAttentionBlock` modules (multi-head attention with position-wise feed-forward and residual connections). This ordering prioritises close-range wrist-camera information while conditioning on global bird's-eye context. Optional spatial self-attention blocks can further refine the fused token.
3. **Temporal transformer.** The per-frame fused tokens, shaped $[B, T, D]$, are augmented with sinusoidal positional encodings and passed through $n_{\text{temporal}} = 2$ self-attention blocks, enabling the model to reason over the full action history within the context window.
4. **Output heads.** Two linear layers read out $\Delta\mathbf{p} \in \mathbb{R}^3$ (translation delta in table frame) and $\Delta\mathbf{r}_{\text{6D}} \in \mathbb{R}^6$ (6D rotation representation) for each time step.

6D rotation representation. Following Zhou et al. [10], we represent rotation as the first two columns of the rotation matrix (6 scalars) and recover the full $\mathrm{SO}(3)$ matrix via Gram–Schmidt orthonormalisation. This representation is continuous everywhere, avoiding the gimbal-lock and discontinuity issues of Euler angles and quaternions.

Loss function. Training minimises a combined loss

$$\mathcal{L} = w_p\,\mathcal{L}_{\text{pos}} + w_r\,\mathcal{L}_{\text{rot}} + w_s\,\mathcal{L}_{\text{smooth}}, \tag{1}$$

where $\mathcal{L}_{\text{pos}} = \|\Delta\hat{\mathbf{p}} - \Delta\mathbf{p}\|_1$ is the ℓ_1 position loss, $\mathcal{L}_{\text{rot}}$ is the geodesic rotation loss

$$\mathcal{L}_{\text{rot}} = \mathbb{E}\left[\arccos\left(\frac{\mathrm{tr}(\hat{R}^\top R_{\text{gt}}) - 1}{2}\right)\right], \tag{2}$$

and $\mathcal{L}_{\text{smooth}}$ penalises the mean squared second finite difference (acceleration) of both $\Delta\hat{\mathbf{p}}$ and $\Delta\hat{\mathbf{r}}_{\text{6D}}$ along the time axis to encourage temporally smooth predictions. Weights are set to $w_p = 1.0$, $w_r = 1.0$, $w_s = 0.01$.

Data collection. Training data consists of multi-camera teleoperation episodes recorded with the so-101 arm. Each episode stores synchronised top-view and wrist-view RGB frames at 30 Hz, together with joint states and commanded actions. Frames are timestamped for synchronisation; the calibration version ID is embedded in episode metadata to enable frame-level extrinsic look-up during training.

Training. We train MVTP-Net for 50 epochs using the AdamW optimiser with cosine-annealing learning rate schedule, initial learning rate $3{\times}10^{-4}$, and weight decay 10^{-4}. Batch size is 32 and the context window is $T = 8$ frames. Automatic mixed-precision (`bf16`) training is enabled on Ampere-class GPUs.

4.5 Point-Cloud Depth Fusion (PointCloudNet)

To supplement the RGB-only MVTP-Net, we develop **PointCloudNet**, a parallel architecture that additionally ingests per-frame depth maps and explicitly back-projected 3-D point clouds. This gives the model direct access to metric geometry, which is important for approach planning and contact detection.

Depth acquisition. Depth maps are either recorded directly alongside RGB at collection time (float32 NumPy arrays, one per frame, stored as `frames/depth/frame_XXXX.npy`) or retrofitted offline via MiDaS monocular depth estimation (`generate_depth_maps.py`). If no depth directory is found the dataset falls back to a zero-depth plane, allowing smoke-tests without a depth sensor.

Point-cloud construction. Each depth frame is back-projected into the camera frame using calibrated intrinsics (f_x, f_y, c_x, c_y) read from `meta.json`. Valid pixels (depth > 0) are converted to XYZ coordinates and uniformly sub-sampled to $N = 2{,}048$ points per frame.

Architecture. PointCloudNet uses four parallel per-frame encoders, all evaluated with time folded into the batch dimension ($B{\cdot}T$):

1. **PointNetEncoder** — processes the $[B{\cdot}T, N, 3]$ point cloud through a per-point MLP ($3 \to 128 \to 256 \to D$, LayerNorm + GELU after each layer), applies *symmetric max-pooling* over the point dimension, and refines the result with a global MLP $\to [B{\cdot}T,\ D]$.
2. **RGB encoder (top)** — three strided convolutions ($3 \to 64 \to 128 \to D$, stride 2), followed by adaptive global average pooling $\to [B{\cdot}T,\ D]$.
3. **RGB encoder (wrist)** — identical architecture as above, with separate weights $\to [B{\cdot}T,\ D]$.
4. **Depth encoder** — mirrors the RGB encoder but with a single-channel input ($1 \to 32 \to 64 \to D$) $\to [B{\cdot}T,\ D]$.

The four feature vectors are concatenated into a $4D$-dimensional vector and passed through a two-layer fusion MLP ($4D \to 2D$, GELU, Dropout; $2D \to D$), producing a single fused frame descriptor $[B{\cdot}T,\ D]$. The descriptor sequence is reshaped to $[B, T, D]$, augmented with sinusoidal positional encodings, and processed by $n_{\text{temporal}} = 2$ self-attention transformer blocks, exactly as in MVTP-Net. Two linear heads produce $\Delta\mathbf{p} \in \mathbb{R}^3$ and $\Delta\mathbf{r}_{\text{6D}} \in \mathbb{R}^6$ per time step. The default feature dimension is $D = 256$ with $n_{\text{heads}} = 8$.

The loss function is identical to that of MVTP-Net (Eq. 1–2): L1 position loss, geodesic rotation loss, and jerk-smoothness penalty with weights $w_p = 1.0$, $w_r = 1.0$, $w_s = 0.01$.

Training. PointCloudNet is trained for 50 epochs with AdamW and cosine-annealing learning rate schedule, initial learning rate 10^{-4}, weight decay 10^{-4}, and batch size 16. The context window is $T = 8$ frames with $N = 2{,}048$ points per frame. Automatic mixed-precision (`bf16`) training is enabled. For high-throughput runs on H200 hardware the batch size is scaled to 128 (learning rate $3{\times}10^{-4}$, 100 epochs, `torch.compile` enabled), and the context window and point-cloud density are increased to $T = 16$ and $N = 4{,}048$ respectively.

4.6 Injection-Spot Estimation and Tangent-Constrained Trajectory

Injection-spot estimator. Given the estimated wrist position $\mathbf{w} = (w_x, w_y, w_z)$ in the table frame, the injection point is computed as a fixed offset

$$\mathbf{b} = (w_x + \delta_x,\ w_y + \delta_y), \tag{3}$$

with $\delta_x = 0.03\,\text{m}$, $\delta_y = -0.02\,\text{m}$. The injection tangent angle is $\phi = \text{atan2}(b_y, b_x)$. To suppress noise, consecutive estimates are blended via exponential moving average with $\alpha = 0.65$, giving a stable injection-spot estimate with confidence score in $[0, 1]$.

Trajectory planner. From the estimated injection point and tangent, the `TangentTrajectoryPlanner` constructs a five-stage motion sequence:

1. **Sleep pose:** retract to home configuration.
2. **Approach:** move to a position $0.04\,\text{m}$ above the injection point, aligned with the tangent yaw.
3. **Tangent align:** rotate in place to match the tangent direction exactly.
4. **Insert along tangent:** translate strictly along the normalised tangent vector by `insert_depth` $= 0.02\,\text{m}$.

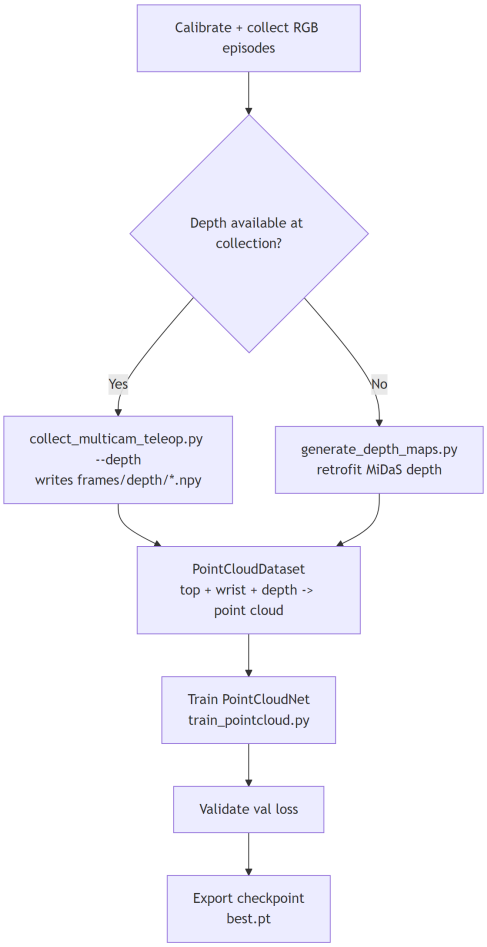

Figure 7: PointCloudNet data pipeline. Depth maps (from sensor or MiDaS) are back-projected into point clouds using calibrated intrinsics. Four parallel encoders process the top RGB, wrist RGB, depth map, and point cloud; a fusion MLP merges them before temporal self-attention and pose-delta regression heads.

5. **Retract:** reverse to the approach pose.

All waypoints are checked against safety bounds ($|x|, |y| \leq 0.60\,\text{m}$, $z \geq 0.01\,\text{m}$) before execution. Maximum Cartesian velocity is capped at $0.03\,\text{m/s}$ and acceleration at $0.08\,\text{m/s}^2$ throughout.

Safety interlocks. An E-stop flag is checked before every motion command and can be triggered from both the hardware button and the web UI. Additionally, each motion primitive checks a perception confidence score against a minimum threshold of 0.50; if confidence falls below this value mid-execution, the controller raises a `SafetyAbort` and the arm returns to the sleep pose.

5 Experiments

5.1 Object Detection

Table 1 summarises the YOLO11n fine-tuning results on the top-camera can detection benchmark.

5.2 Wrist Pose Estimation

Figure 10 compares MVTP-Net predicted positions against ground-truth end-effector positions across the evaluation set, and Figure 11 shows the distribution of per-axis and Euclidean endpoint errors.

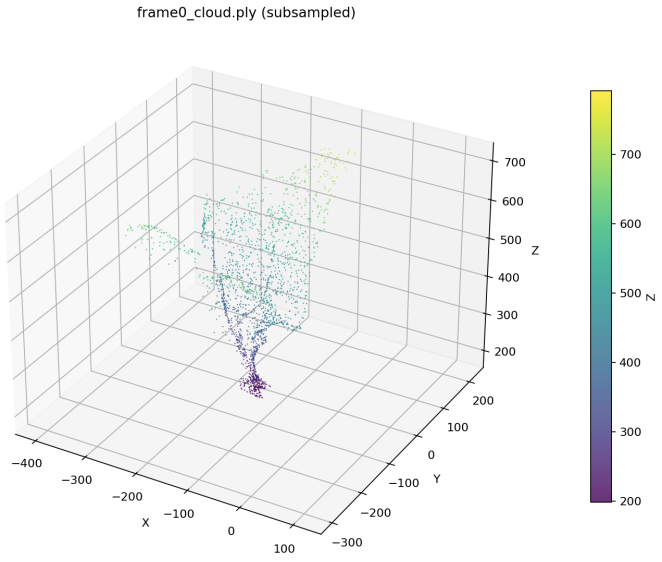

Figure 8: Example point cloud back-projected from a single top-camera depth frame after calibration. Valid-depth pixels (depth > 0) are sub-sampled to $N = 2{,}048$ points and expressed in the camera frame (metres).

Table 1: YOLO11n can detection performance (epoch 80, validation set).

Model	Precision	Recall	mAP@50	mAP@50:95
YOLO11n (fine-tuned)	89.7%	89.8%	89.6%	35.9%

5.3 Gaussian-splat-based mapping

Figure 3 shows the final map the kiwi-platform uses for navigation. VLA-model trained on ACT-policy was used to navigate through the map based on UI-provided instructions.

6 Conclusion

We have presented MANTIS-4D, an integrated autonomous robotic system combining Gaussian-splat-based mapping, multi-view temporal pose estimation, YOLO-based target detection, and tangent-constrained manipulation. The modular architecture decouples perception, planning, and actuation while sharing a common table-frame coordinate system enforced by a versioned multi-camera calibration pipeline. The system is designed for safe, supervised deployment in novel rooms, exposing every inference decision to the operator via a real-time web UI with a hardware E-stop.

Future work will focus on tightening endpoint accuracy through higher-frequency calibration and online adaptation of the injection-spot estimator, extending the Gaussian-splat map to support semantic object queries for autonomous table-finding, and replacing the fixed-offset injection estimator with a learned model trained on depth-enhanced point-cloud data.

7 Use Of AI

This overview was AI generated based on our work from the Github-repository.

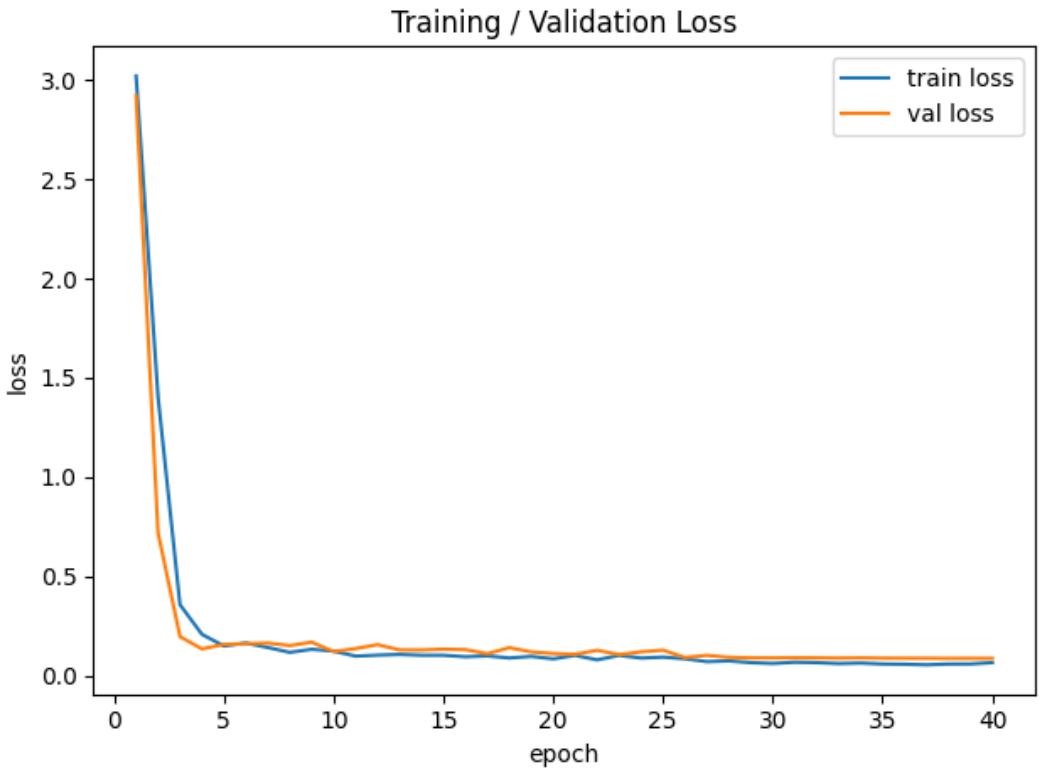

Figure 9: Loss for MVTP-Net

References

[1] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. In *ACM Transactions on Graphics (SIGGRAPH)*, volume 42, pages 139:1–139:14, 2023.

[2] Antoni Rosinol, John J. Leonard, and Luca Carlone. NeRF-SLAM: Real-time dense monocular SLAM with neural radiance fields. *arXiv preprint arXiv:2210.13641*, 2023.

[3] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. SplaTAM: Splat, track & map 3D gaussians for dense RGB-D SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[4] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian splatting SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[5] Timothy Chen, Ola Shorinwa, Joseph Butler, Javier Zeng, Jonathan Bohren, Philip Dames, and Mac Schwager. Splat-nav: Safe real-time robot navigation in Gaussian splatting maps. *arXiv preprint arXiv:2403.02751*, 2024.

[6] Yann Labbé, Lucas Manuelli, Arsalan Mousavian, Stephen Tyree, Stan Birchfield, Jonathan Tremblay, Justin Carpentier, Nicolas Mansard, Tomás Lozano-Pérez, Michael Kaess, et al. MegaPose: 6d pose estimation of novel objects via render & compare. In *Conference on Robot Learning (CoRL)*, Proceedings of Machine Learning Research, 2023.

[7] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. FoundationPose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[8] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. CosyPose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision (ECCV)*, 2020.

[9] Apurva Beedu, Hanan Alamri, and Irfan Essa. Video based object 6D pose estimation using transformers. *arXiv preprint arXiv:2201.09232*, 2022.

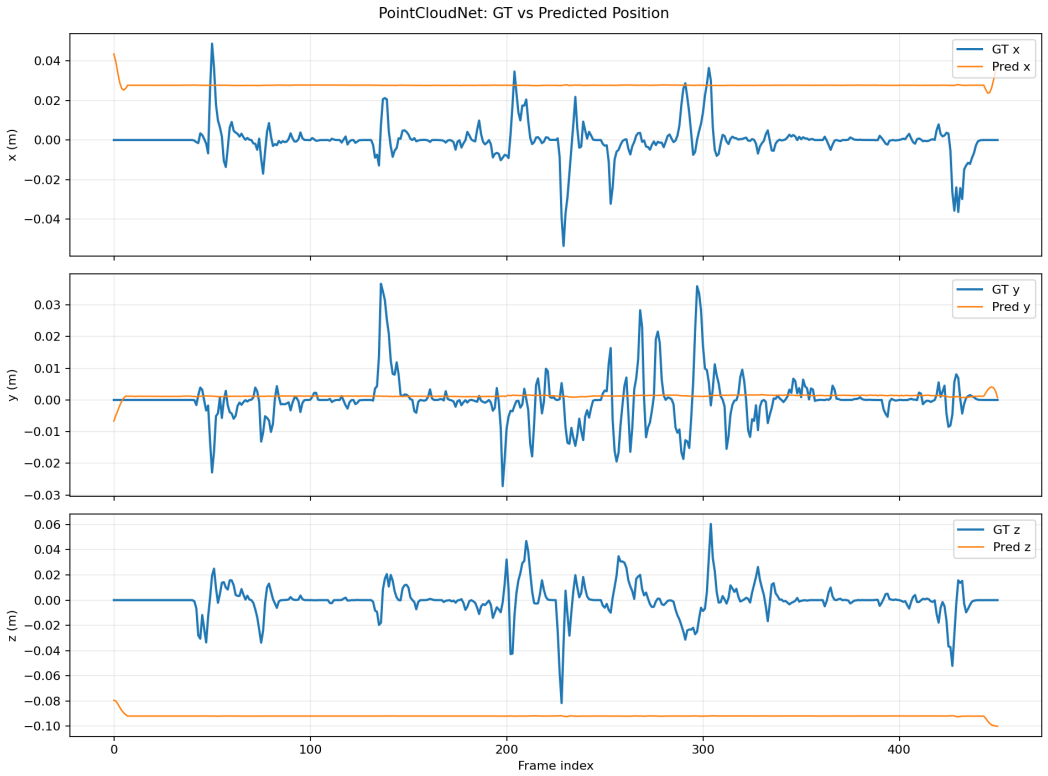

Figure 10: Ground-truth vs. predicted end-effector positions produced by MVTP-Net on the held-out evaluation set. Each point corresponds to one time step; diagonal alignment indicates accurate tracking along each Cartesian axis.

[10] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5745–5753, 2019.

[11] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635, 2011.

[12] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

[13] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[14] Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile ALOHA: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[16] Ultralytics. Ultralytics YOLO11, 2024. Released September 2024.

[17] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

[18] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development, 2023.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An

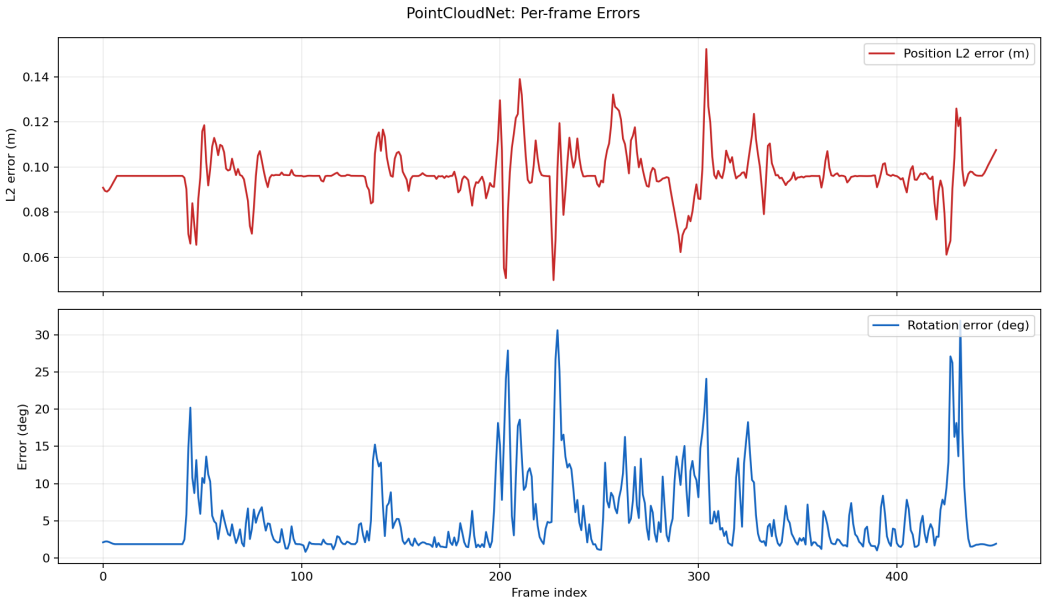

Figure 11: Distribution of per-axis (x, y, z) and Euclidean position prediction errors (in metres) on the evaluation set.

image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations (ICLR)*, 2021.