Depth-Map Denoising and Restoration for RGB-D 3-D Reconstruction and Tracking: Robot-Arm Platform, Sensor-Noise Analysis, and Tracking

Chuhan ZHANG — 999016355 — Group 2 — DISP Final Project
Member A: Chuhan ZHANG
Member B: Jianting WU
Member C: Junzhe XU

***Abstract*—Depth maps from consumer RGB-D cameras are noisy and full of holes, and this corruption flows directly into 3-D reconstruction and camera tracking. Our group's research question is: *which classical denoising/restoration filter cleans a real depth map best, and does the answer still hold after the full 3-D reconstruction and tracking pipeline?* We compare six classical filters (mean, Gaussian, median, frequency-domain Gaussian and Butterworth low-pass, and bilateral) on a real dataset that we captured ourselves: a desktop battery scene recorded by an Intel RealSense D435 carried by an OpenManipulator-X robot arm along a scan path planned in task space, executed through inverse kinematics, and referenced through calibrated forward kinematics. We first verify Nguyen *et al.*'s [2] distance-dependent noise model on our own D435 (full fit in Section IV-A). We then score every filter with PSNR, SSIM, a Sobel edge score, a 3-D point RMSE, and finally with camera-tracking error against two references (an oracle trajectory and the arm's forward kinematics). Controlled tests use three synthetic noise types we define: Gaussian (distance-growing, operating point $\sigma_0 \approx 18$ mm), salt-and-pepper (impulse / flying pixels), and speckle (multiplicative), plus a noise-level sweep and missing-data dropouts. We find three things: (1) at the operating point the bilateral filter is the best default; (2) no filter wins everywhere; median under impulse, mean under speckle, bilateral under Gaussian; and (3) the 2-D ranking carries over to tracking (Experiment B), with an order of magnitude between matched and mismatched filter–noise pairs, while real-noise Experiment A is dominated by my FK reference floor ($\sim$17 mm). In my report the deep analysis is Methodology and Results for the platform and tracking: hardware and scan-path formulas (FK/IK, time sync), the measured noise model, hybrid RGB-D odometry, and Experiments A/B; 2-D/3-D figures appear for the full story but companion sections carry the detailed derivations.**

***Index Terms*—RGB-D sensing, depth denoising, robot arm, forward kinematics, visual odometry, RealSense D435**

I. Introduction

THE project is a *paper replication with research extension*. It has three objectives:

1) **Replication.** Reproduce the classical "add known noise $\rightarrow$ restore $\rightarrow$ score with PSNR/SSIM" evaluation from the image-restoration literature [5] on depth captured with a real Intel RealSense D435, and verify KinectFusion's [1] choice of the bilateral filter as depth preprocessing, after first validating Nguyen *et al.*'s [2] distance-dependent noise model on our own sensor.
2) **Extension in 2-D.** Broaden the evaluation to several noise levels, three noise types, and missing-data conditions; quantify the smoothing-versus-edge trade-off; and check that the conclusions hold on more than one scene.
3) **Extension to 3-D.** Close the loop: back-project restored depth to 3-D for a point-distance RMSE, fuse the sequence into a mesh, and measure how each filter changes camera-tracking accuracy against an oracle trajectory and against the calibrated forward kinematics (FK) of the robot arm that carries the camera.

My personal responsibility covers the platform-dependent parts of all three objectives: building the capture rig; planning and executing the scan; calibrating the FK reference used in Objective 3; verifying the noise model that grounds Objective 2's synthetic-noise protocol; and running the closed-loop tracking comparison that serves as the final test of Objective 3.

Scope. We stay within the classical, interpretable filter family taught in the DISP course [12]. Learning-based restoration is reviewed but not implemented. Absolute ground-truth poses (motion capture) are out of scope. Instead, I use the oracle- and FK-based protocols of Sec. III-I.

II. Related Work

A. RGB-D Depth Sensing and Its Noise

Active RGB-D cameras measure depth in two principal ways. Structured-light systems (Kinect v1) project a known pattern and decode depth from its deformation. The Intel RealSense D435 uses active-stereo triangulation instead. An infrared projector paints texture on the scene; two infrared imagers then triangulate correspondences. Triangulation error grows quadratically with distance. The finite disparity resolution also makes axial noise distance-dependent. Nguyen *et al.* [2] measured Kinect noise against a planar target over many distances and angles and fit *axial* noise as a quadratic function of depth, and *lateral* noise as a linear function of depth, additionally diverging at grazing angles. They then plugged this model into KinectFusion's TSDF weighting and ICP correspondence rejection and obtained visibly cleaner reconstructions and more robust tracking on fine structures. Two messages from that paper guide our study: (i) depth noise is *not* white; it depends on distance and incidence

angle. Any synthetic-noise experiment should respect that structure, which is exactly what I verify on our own sensor in Sec. III-E; (ii) the right place to judge a depth-processing choice is the *downstream* reconstruction and tracking quality, which is exactly what my closed-loop experiments (Sec. IV-D) test. A *Sensors* 2025 study [10] repeats the exercise on a modern structured-light sensor mounted, like ours, on a robot arm. Feeding a measured noise model into KinectFusion-class pipelines improves both tracking and reconstruction. This is direct evidence that the comparison I run in Sec. IV-D is measuring something that matters.

B. Classical Denoising and Restoration Filters

Standard texts [5] organize denoising into spatial masks (mean, Gaussian, median) and frequency-domain transfer functions (Gaussian and Butterworth low-pass). Linear filters are least-squares optimal for additive noise but blur edges; the median filter resists impulse noise because a single outlier cannot dominate the sorted middle value; frequency low-pass filters remove all high-frequency content, noise and edges alike. The bilateral filter of Tomasi and Manduchi [3] adds a range Gaussian to the spatial one,

$$g(p) = \frac{1}{W_p} \sum_{q \in \mathcal{N}(p)} f(q)\, e^{-\frac{\|p-q\|^2}{2\sigma_s^2}}\, e^{-\frac{(f(p)-f(q))^2}{2\sigma_r^2}}, \qquad (1)$$

so it smooths within a surface while keeping sharp depth discontinuities intact—why KinectFusion adopted it, and why we see it winning at the sensor's operating point. PSNR, SSIM [4], and a Sobel edge score are the 2-D metrics used throughout; the filter study defines and discusses them in depth.

C. Dense 3-D Reconstruction and Tracking

KinectFusion [1] reconstructs dense surfaces in real time from a moving depth camera by alternating four stages (Fig. 1): surface measurement (bilateral filtering, vertex and normal maps), pose estimation (point-to-plane ICP against the raycast model), reconstruction update (TSDF weighted average of *raw* depth), and surface prediction (raycasting). The two design choices that matter most for this project—filter for tracking, fuse raw—are implemented and tested in detail elsewhere; what matters for my report is that tracking consumes *normals*, so the filter that best preserves edges/normals should also track best, which is exactly the hypothesis my closed-loop experiments test.

D. Learning-Based Depth Restoration

Because clean depth ground truth is scarce, recent methods are self-supervised or diffusion-based [6]–[9] and restore depth on transparent and reflective materials where classical filters fail entirely. They cost far more compute and are harder to interpret; our classical baseline (this project) is the kind of reference these methods are typically compared against. This does not affect my platform or tracking results directly, but it frames why the reconstruction pipeline discusses them as future work.

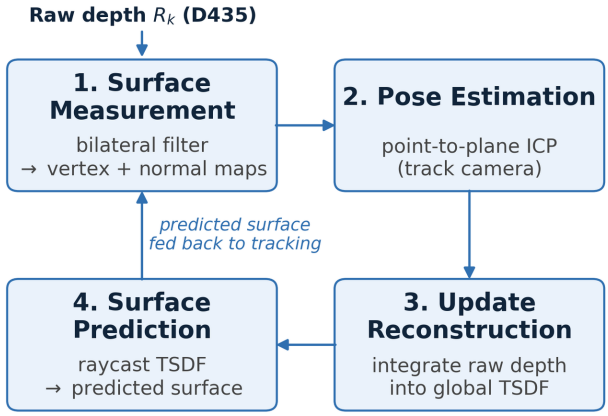


Fig. 1: KinectFusion-style pipeline used in our reconstruction and tracking experiments (after [1], used in this project): bilateral filtering feeds tracking normals; raw depth is fused into the TSDF.

E. Summary and Gap

The literature establishes sensor-specific noise models, a classical filter toolbox with known trade-offs, and evidence that depth quality affects tracking. What is less documented is a single controlled comparison that follows the same filters from 2-D metrics through 3-D geometry to tracking accuracy on one real, robot-carried dataset—which is the gap my platform exists to fill.

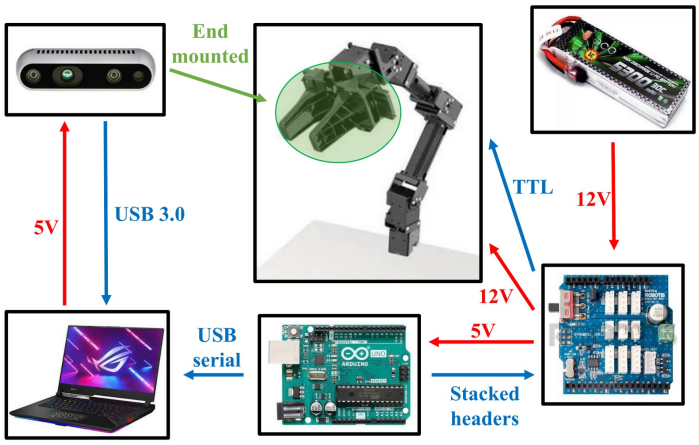
III. Methodology

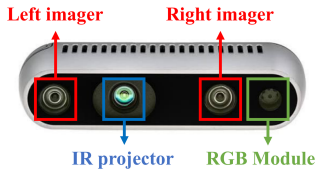
A. Study Design

The replication component reproduces the classical restoration protocol of [5] and KinectFusion's bilateral pre-filtering choice [1]; the extension component adds the 3-D metric, the multi-condition noise study, the scene-diversity check, and the closed-loop tracking evaluation. All primary numbers come from one self-collected robot-arm dataset that I built and captured; two auxiliary scenes captured earlier in the project support downstream checks to confirm the conclusions generalize.

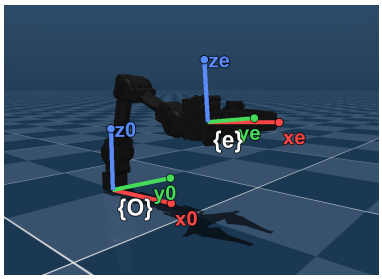
B. Hardware Platform

Fig. 2 shows the platform I designed and assembled (left: wiring and power/signal paths). An Intel RealSense D435 streams 640×480 color and depth at 30 Hz with every SDK post-processing filter *disabled*, so the rawest (noisiest) depth is recorded together with the factory intrinsics (f_x, f_y, c_x, c_y). The D435 layout and the calibrated MuJoCo model used for FK are on the right (panels a–b). The camera is mounted on the end effector of a ROBOTIS OpenManipulator-X: four Dynamixel XM430 servos driven by an Arduino Uno through a DYNAMIXEL Shield. Power is split by design: an 11.1 V LiPo pack powers the shield and the servo daisy-chain, while the camera draws 5 V over its USB 3.0 link to the PC; the Arduino receives joint targets over USB serial. Separating





(a) RealSense D435 sensor layout.



(b) OpenManipulator-X in MuJoCo with base frame $\{O\}$ and end-effector frame $\{e\}$, calibrated to the physical home pose.

Fig. 2: My capture platform. **Left:** hardware wiring (red: 12 V LiPo to shield/servos, 5 V USB to camera; blue: serial, TTL, USB 3.0). **Right:** (a) D435 sensor; (b) calibrated MuJoCo FK model.

motor power from sensor power avoids brown-outs during rapid joint motion, and mounting the camera with a re-alignable clamp (rather than a fixed screw) let me iterate on the mount without re-wiring, at a small cost in mechanical rigidity that shows up as high-frequency jitter in the noise measurement of Sec. III-E.

C. Scan-Path Planning: IK Execution, FK Reference

This is the part of the pipeline that is entirely my responsibility, and it follows a four-step procedure.

a) Kinematic model.: The OpenManipulator-X is a 4-DOF serial chain (one base yaw and three pitch joints). Following the ROBOTIS `robotis_manipulator` library—the same chain as in Fig. 2b—each link transform is a fixed translation $\mathbf{a}_i$ followed by a rotation about the local Z (base) or Y (shoulder/elbow/wrist) axis:

$$T_{i-1\to i}(q_i) = \mathrm{Trans}(\mathbf{a}_i)\,\mathrm{Rot}_{\mathrm{axis}_i}(q_i), \tag{2}$$

with $\mathbf{a}_1 = (12, 0, 17)\,\mathrm{mm}$, $\mathbf{a}_2 = (0, 0, 59.5)\,\mathrm{mm}$, $\mathbf{a}_3 = (24, 0, 128)\,\mathrm{mm}$, $\mathbf{a}_4 = (124, 0, 0)\,\mathrm{mm}$, and a fixed tool offset $\mathbf{a}_{\mathrm{tool}} = (126, 0, 0)\,\mathrm{mm}$ to the nominal gripper center. Rotations follow the ROBOTIS convention (Z for joint 1, Y for joints 2–4):

$$R_z(q) = \begin{bmatrix} \cos q & -\sin q & 0 \\ \sin q & \cos q & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad R_y(q) = \begin{bmatrix} \cos q & 0 & \sin q \\ 0 & 1 & 0 \\ -\sin q & 0 & \cos q \end{bmatrix}. \tag{3}$$

Each link is a homogeneous transform $\mathbf{T}_{i-1\to i} = \begin{bmatrix} R_{\mathrm{axis}_i}(q_i) & \mathbf{a}_i \\ \mathbf{0}^\top & 1 \end{bmatrix}$, and the end-effector pose is

$$\mathbf{T}_{0\to e}(\mathbf{q}) = \Big(\prod_{i=1}^{4} \mathbf{T}_{i-1\to i}\Big) \begin{bmatrix} I & \mathbf{a}_{\mathrm{tool}} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{4}$$

Eq. (11) extracts the translation column $\mathbf{p}_e$ from $\mathbf{T}_{0\to e}$. The same constants appear in the MuJoCo model and in my Arduino FK verification sketch, so simulation, commanded motion, and the FK reference stay consistent once joint conventions match.

The RealSense D435 (nominal mass 72 g) is clamped ∼50 mm in front of the wrist flange; its optical center $\mathbf{p}_c$ is not the gripper center $\mathbf{p}_e$ in Eq. (11) but

$$\mathbf{p}_c(t) = R_{0\to e}\big(\mathbf{q}(t)\big)\,\boldsymbol{\ell} + \mathbf{p}_e(t), \tag{5}$$

where $\boldsymbol{\ell}$ is the lever arm in the end-effector frame ($\mathbf{T}_{0\to c} = \mathbf{T}_{0\to e}\,\mathrm{Trans}(\boldsymbol{\ell})$). I treat $\boldsymbol{\ell} \approx \mathbf{0}$ in the main FK reference because I never ran a dedicated hand–eye calibration; the constant offset this introduces is part of the FK bound in Sec. IV-D. Servo readings are mapped to model angles by

$$q_i^{\mathrm{model}} = q_i^{\mathrm{stored}} - \delta_i, \tag{6}$$

with δ_i in Table I. The large corrections on joints 2 and 3 are consistent with an extra static load: camera, mount (∼30 g aluminium), and USB cable pull add a wrist moment that deflects the shoulder and elbow before the encoders read zero. A first-order static balance at joint i gives

$$\tau_i^{\mathrm{load}} \approx \sum_{k\ge i} m_k\, g\, \frac{\partial z_k}{\partial q_i} + m_{\mathrm{cam}}\, g \left\| \Big(\frac{\partial \mathbf{r}_{\mathrm{cam}}}{\partial q_i}\Big)_{\perp} \right\|, \tag{7}$$

where z_k is link height in the world frame and $\mathbf{r}_{\mathrm{cam}}$ follows Eq. (5). The XM430 servos run in position mode with internal

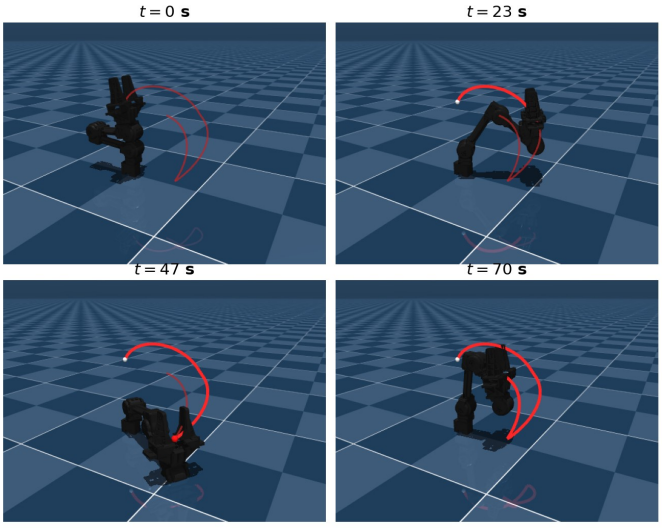


Fig. 3: MuJoCo replay of the 70 s scan path: four snapshots along the smooth joint-space profile (20 Hz) through three taught waypoints.

PID, so I do not identify τ_i per joint; Eq. (7) only explains why load-dependent sag concentrates on joints 2–3 and why FK remains a *bound*, not ground truth.

b) Step 1 — task-space scan path.: The scan is specified where the task actually lives: a smooth end-effector sweep $\mathbf{p}(t)$ over the battery desktop in Cartesian space, chosen so the camera keeps the target in view while the viewing direction changes enough to exercise the tracker. Joint commands are derived *from* this motion plan rather than being chosen pose by pose, so the plan is reusable if the arm's kinematic parameters ever change. Compared with a hand-held scan (Scene 2 in the hand-held capture) or a fixed camera, eye-in-hand motion on a known $\mathbf{q}(t)$ gives three advantages we exploit: (i) **repeatability**—every run follows the same 70 s profile through three taught waypoints (Fig. 3), so filter and tracking comparisons are not confounded by operator motion; (ii) **controlled dynamics**—peak joint speed stays below $45°$/s with Dynamixel *Profile Velocity* 25 and *Profile Acceleration* 10, limiting motion blur and keeping consecutive RGB-D frames within the 30 Hz odometry capture rate; and (iii) **view diversity**—the arc spans roughly $60°$ of base yaw and $\pm 40°$ of elevation change while keeping the desktop in the D435 field of view, which multi-view ICP needs without re-entering joint limits.

c) Step 2 — inverse kinematics and MuJoCo verification.: The task-space path is converted to a time-varying joint trajectory $\mathbf{q}(t)$ by the OpenManipulator-X IK solver (a damped least-squares Jacobian iteration built into the ROBOTIS firmware), subject to the four joint limits of the XM430 chain, and verified in the official ROBOTIS MuJoCo model (Fig. 3) before touching hardware, which catches self-collisions and joint-limit violations for free. For each Cartesian target $\mathbf{p}^\star$ on the path the IK updates

$$\Delta\mathbf{q} = J(\mathbf{q})^\top \big(JJ^\top + \lambda^2 I_3\big)^{-1} \Delta\mathbf{p}, \quad J_{j,i} = \frac{\partial p_j}{\partial q_i}, \tag{8}$$

with damping $\lambda \approx 0.08$ for stability near singular postures (verified against my own standalone IK implementation). Between waypoints the Arduino streams joint targets at 20 Hz ($\Delta t_{\mathrm{cmd}} = 50$ ms) with piecewise-linear interpolation

$$\mathbf{q}(t) = (1-\alpha)\mathbf{q}_k + \alpha\mathbf{q}_{k+1}, \quad \alpha = \frac{t - t_k}{t_{k+1} - t_k}, \tag{9}$$

while the servos apply their internal trapezoidal velocity profiles. The simulation replays the same continuous 70-s joint-space profile at 20 Hz that is later executed on hardware, with no intermediate stops.

d) Step 3 — execution and temporal storyboard.: At capture time the Arduino streams the interpolated joint targets over the DYNAMIXEL serial bus at 20 Hz while the RealSense records RGB-D independently at 30 Hz. Because the two streams are not hardware-triggered from a shared clock, I align them in software: both logs carry host-computer timestamps, and I solve for a single scalar offset Δt that minimizes the FK-vs-frame alignment error at a few visually identifiable pose changes, then hold Δt fixed for the whole sequence. For RGB-D frame k with host timestamp $t_{\mathrm{rel},k}$,

$$\begin{aligned} t_{\mathrm{motion},k} &= t_{\mathrm{rel},k} - t_{\mathrm{countdown}} + \Delta t, \\ \mathbf{q}_k &= \mathbf{q}(t_{\mathrm{motion},k}), \\ \mathbf{p}_k^{\mathrm{FK}} &= \mathbf{p}_e\big(\mathbf{q}_k\big), \end{aligned} \tag{10}$$

where $\mathbf{q}(t)$ follows Eq. (9) and $\mathbf{p}_e$ from Eqs. (4)–(11). The arm moves slowly enough (peak joint speed $< 45°$/s) that a timing error of even 100 ms only shifts the FK position estimate by a few millimeters, well below the tracking floor discussed in Sec. IV-D. Fig. 4 aligns, at four instants $t = 10.0, 33.4, 56.7, 70.0$ s, the physical desk, the RGB view, and the raw depth.

e) Step 4 — forward kinematics as tracking reference.: The commanded joint trajectory maps back to end-effector poses through Eqs. (4)–(11):

$$\mathbf{p}(t) = \mathbf{p}_e\big(\mathbf{q}(t)\big) = \mathbf{T}_{0\to e}\big(\mathbf{q}(t)\big)[0{:}3,\, 3], \tag{11}$$

yielding a reference path for the visual odometry. The Dynamixel zero positions depend on how the servo horns were assembled and do *not* coincide with the official model zeros: a servo reports $q_i = 0$ at whatever angle its horn happened to be glued, not at the model's $q_i = 0$. I calibrated the per-joint offsets in Table I by physically driving the arm to a recognizable stored home pose and manually searching offsets in the MuJoCo model until the simulated pose matched both the photograph and the manufacturer's home-pose specification (Fig. 2b). This is a one-time, one-pose calibration, so it removes the constant part of the mismatch but not any load-dependent sag; the two lowest joints, which carry almost the whole arm's weight, receive the largest offsets, consistent with a heavier static load deflecting the horn further before the internal encoder reads zero. The FK path is then time-aligned to the frames with the single shared offset Δt from Step 3, searched once and used for every filter/tracking combination. Because FK interpolates commanded waypoints (no joint encoders) and the hand–eye transform between flange and optical center is not calibrated, FK *bounds* rather than measures absolute tracking error—the central limitation I quantify in Sec. IV-D.

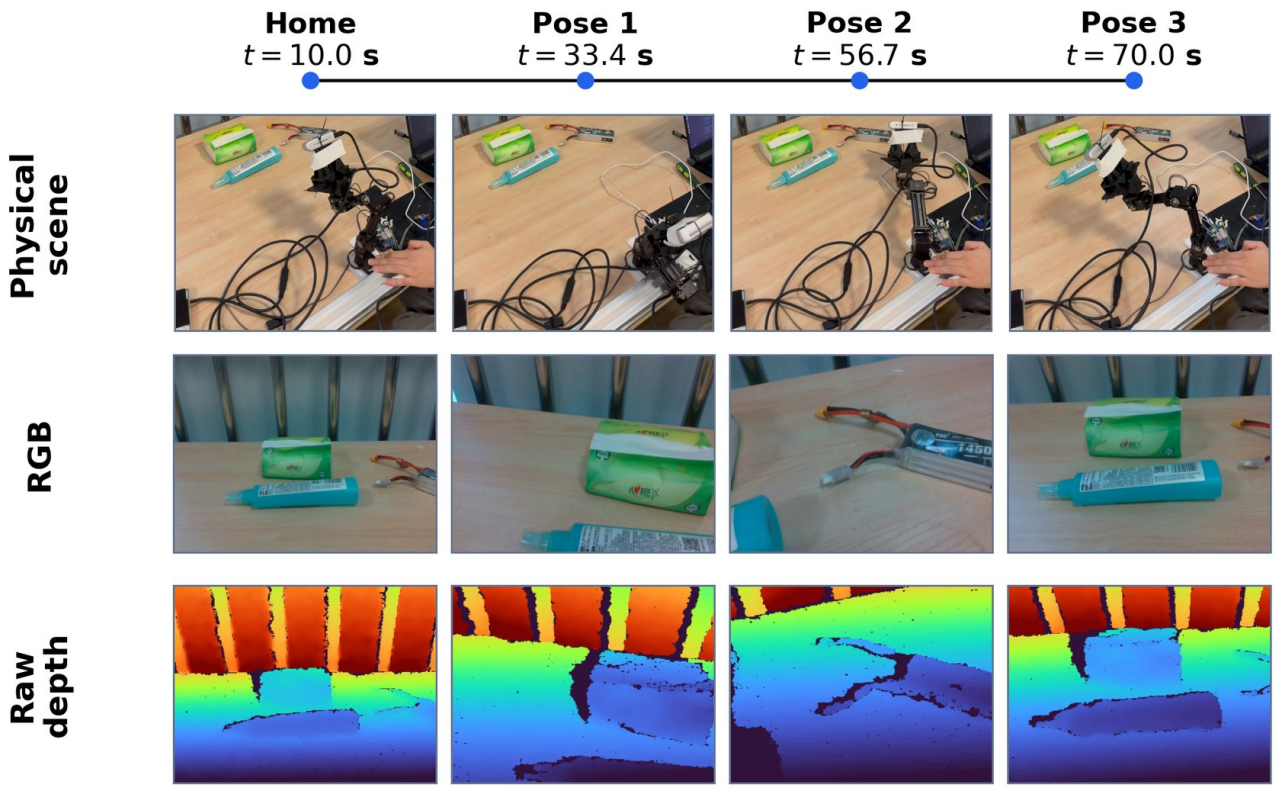


Fig. 4: Execution timeline of the capture I ran. Each column shows the same instant: physical scene, RGB, and raw depth.

TABLE I: Per-joint calibration offsets applied to the servo-reported angle to obtain the MuJoCo model angle.

Joint	1 (base)	2 (shoulder)	3 (elbow)	4 (wrist)
Offset δ_i (°)	+87.4	−16.6	+96.6	+3.0

D. Datasets

One 70-s motion was recorded and cropped to the steady 60-s segment, giving 1798 RGB-D frames of the desktop battery scene. This sequence, which I captured, provides every primary number in this report: probe frames for the restoration study, the static countdown segment for my noise measurement, and the full sweep for the reconstruction and for my tracking comparison. Two auxiliary scenes captured earlier with the same sensor are reused by the filter study for the scene-diversity check: a sofa/guitar scene (500 frames, hand-held) and a hole-fill demo on the robot-arm battery frames (both analyzed in the companion reports).

E. Sensor-Noise Measurement and the Study's Noise Model

Computing PSNR/SSIM requires a clean reference, which real noise cannot provide. The group's protocol therefore (i) hole-fills the chosen real frame and applies a mild bilateral pass to obtain a dense *clean reference*, and (ii) adds *known* noise to form the noisy input, growing with depth as

$$\sigma(Z) = \sigma_0 + \alpha Z, \tag{12}$$

following the distance-dependent behavior measured by Nguyen *et al.* [2]. This is not an assumption I take on faith—I *verified it on our own sensor*. During the countdown segment of the recording the arm holds its home pose, giving 210 frames of a static scene from a static camera; I compute the per-pixel temporal standard deviation of depth, bin it by distance, and fit both the linear model above and a Nguyen-style quadratic $\sigma(Z) = a + bZ + cZ^2$ for comparison, using only the static frames so the measurement is independent of the arm's own vibration. The fitted operating point $\sigma_0 \approx 18\,\text{mm}$ sits deliberately at the top of the measured range, so every filtering experiment in the project stresses the filters slightly harder than the sensor's average noise, and the same controlled noise is reused in the 3-D tracking experiments for a consistent comparison.

F. Filters and 2-D Metrics

All spatial filters use 5×5 windows: mean, Gaussian ($\sigma = 1.4$), median, and bilateral (1) with $\sigma_s = 7\,\text{px}$ and $\sigma_r = 40\,\text{mm}$. The frequency-domain Gaussian and Butterworth low-pass filters (order $n = 2$) use the same cutoff D_0 chosen to match the spatial smoothing strength. PSNR, SSIM, and the Sobel edge-preservation score are computed on the study frame (frame 300) and, for stability, averaged over six probe frames spread along the sequence I captured. Companion documentation gives the full rationale for these choices.

G. Extension 1: 2-D to 3-D Back-Projection Metric

Each depth pixel (u, v) with metric depth $Z_{uv} > 0$ back-projects to a 3-D point in the camera frame via the pinhole

model recorded with every frame:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (u - c_x)\, Z_{uv}/f_x \\ (v - c_y)\, Z_{uv}/f_y \\ Z_{uv} \end{bmatrix}, \tag{13}$$

using factory intrinsics (f_x, f_y, c_x, c_y). The reconstruction pipeline stacks these points across frames (after the per-frame odometry pose $\mathbf{T}_k$ is known) to form reference and restored clouds; the resulting **3-D RMSE**

$$\mathrm{RMSE}_{3\mathrm{D}} = \sqrt{\frac{1}{|\mathcal{M}|} \sum_{(u,v)\in\mathcal{M}} \left\| \mathbf{X}^{\mathrm{rest}}_{uv} - \mathbf{X}^{\mathrm{ref}}_{uv} \right\|_2^2} \tag{14}$$

over valid pixels $\mathcal{M}$ is the metric (millimeters). I use it downstream only as a sanity check that my tracking floor is of a comparable order of magnitude to the raw per-point geometric error.

H. Extension 2: Noise Conditions and Scene Diversity

The synthetic noise is varied along three axes (level sweep, three noise types, and random dropouts) and the full 2-D protocol is repeated on two additional scenes; this is the extension, and I reuse its noise-type conditions unchanged in my own Experiment B (Sec. III-I) so that the tracking comparison is apples-to-apples with the 2-D comparison.

I. Extension 3: Closed-Loop Reconstruction and Tracking

This is the extension I own. Following KinectFusion (Fig. 1), each new RGB-D frame k is registered to frame $k-1$ with Open3D's *hybrid* RGB-D odometry: a photometric term on intensity I plus a geometric point-to-plane term on back-projected vertices $\mathbf{V}_k(u)$ from Eq. (13),

$$\begin{aligned} E_k(\Delta\mathbf{T}) = &\sum_{u\in\mathcal{O}} \Big(\nabla I^{\top}_{k-1} \big(\pi(\Delta\mathbf{T}\,\mathbf{V}_k(u)) - \pi(\mathbf{V}_{k-1}(u)) \big) \Big)^2 \\ &+ \sum_{u\in\mathcal{O}} \Big[\big(\Delta\mathbf{T}\,\mathbf{V}_k(u) - \hat{\mathbf{V}}_{k-1} \big) \cdot \hat{\mathbf{N}}_{k-1} \Big]^2, \end{aligned} \tag{15}$$

where $\pi(\cdot)$ projects a 3-D point to the image plane, $\mathcal{O}$ is the set of valid correspondences, and $\hat{\mathbf{V}}_{k-1}, \hat{\mathbf{N}}_{k-1}$ are the model point/normal from the previous frame. The incremental transform $\Delta\mathbf{T}_k \in SE(3)$ is found by Gauss–Newton on Eq. (15) and composed into the camera trajectory

$$\begin{aligned} \mathbf{T}_k &= \mathbf{T}_{k-1}\, \Delta\mathbf{T}_k, \\ \mathbf{p}^{\mathrm{slam}}_k &= \mathbf{T}_k[0{:}3,\, 3], \end{aligned} \tag{16}$$

starting from $\mathbf{T}_0 = I_4$. Only the depth map is pre-filtered in my experiments; the color channel is shared across all filter variants so that differences in $\mathbf{p}^{\mathrm{slam}}_k$ come from the depth term in Eq. (15), not from accidental RGB changes. The point-to-plane term alone can be written as

$$E_d(\Delta\mathbf{T}) = \sum_{u} \big[\big(\Delta\mathbf{T}\,\mathbf{V}_k(u) - \hat{\mathbf{V}}_{k-1} \big) \cdot \hat{\mathbf{N}}_{k-1} \big]^2, \tag{17}$$

which is why surface normals—and therefore depth smoothing—matter for tracking. Fusion maintains a weighted running average per TSDF voxel,

$$F_k = \frac{W_{k-1} F_{k-1} + w\, F_{R_k}}{W_{k-1} + w}. \tag{18}$$

a) Trajectory error metrics.: Visual odometry has a gauge freedom (unknown global Sim(3)/SE(3)), so before comparing $\mathbf{p}^{\mathrm{slam}}_k$ to a reference I apply Umeyama's closed-form Sim(3) alignment [11]:

$$\hat{\mathbf{p}}_k = s\,\mathbf{R}\,\mathbf{p}^{\mathrm{slam}}_k + \mathbf{t}, \quad (s, \mathbf{R}, \mathbf{t}) = \arg\min_{s,\mathbf{R},\mathbf{t}} \sum_k \left\| \hat{\mathbf{p}}_k - \mathbf{p}^{\mathrm{ref}}_k \right\|_2^2, \tag{19}$$

with $\mathbf{p}^{\mathrm{ref}}_k$ either my FK path $\mathbf{p}^{\mathrm{FK}}_k$ from Eq. (10) (Experiment A) or the oracle path from odometry on clean reference depth (Experiment B). The **absolute trajectory error** (ATE) is then

$$\mathrm{ATE} = \sqrt{\frac{1}{N} \sum_{k=1}^{N} \left\| \hat{\mathbf{p}}_k - \mathbf{p}^{\mathrm{ref}}_k \right\|_2^2}, \tag{20}$$

reported in millimeters. For controlled-noise comparisons I additionally use **relative pose error** over Δ frames,

$$\mathrm{RPE}(\Delta) = \sqrt{\frac{1}{N-\Delta} \sum_{k=1}^{N-\Delta} \left\| \big(\hat{\mathbf{p}}_{k+\Delta} - \hat{\mathbf{p}}_k \big) - \big(\mathbf{p}^{\mathrm{ref}}_{k+\Delta} - \mathbf{p}^{\mathrm{ref}}_k \big) \right\|_2^2}, \tag{21}$$

which measures local drift and is less sensitive to a single global misalignment than ATE. I use $\Delta = 10$ frames in Experiment B (≈ 0.33 s at 30 Hz). Equations (15)–(21) are why I designed two experiments rather than one: tracking consumes *normals*, and I need both a real-world test and a controlled test to separate the filter's effect from confounds in my own reference trajectory.

- **Experiment A (real noise, FK reference).** The raw recorded depth (real sensor noise) is pre-processed by eight variants (raw, the six filters, and hole-fill+bilateral) and tracked; ATE is measured against my calibrated FK trajectory of Sec. III-C.
- **Experiment B (controlled noise, oracle reference).** A dense clean reference depth is built for every frame and tracked, yielding an *oracle* trajectory that is exact by construction. Each of the noise types is then added, each filter applied, the identical odometry re-run, and RPE measured against the oracle. This isolates the filter's effect on tracking from any error in my own FK reference.

Both experiments share the same odometry code and differ only in the input depth and the reference trajectory, so that any difference in the results can be attributed to the depth pre-processing rather than to incidental implementation differences.

J. Software and Reproducibility

All code is Python: `pyrealsense2` (capture), OpenCV (filtering, metrics), NumPy, Matplotlib, Open3D (odometry, TSDF fusion, meshing), and MuJoCo with the official ROBOTIS model (arm calibration and simulation, my responsibility). The Arduino sketch driving the Dynamixel waypoint motion is also mine.

IV. Results and Analysis

Unless noted, metrics use $\sigma_0 \approx 18$ mm on frame 300 of the arm sequence I captured. Figures for every stage of the

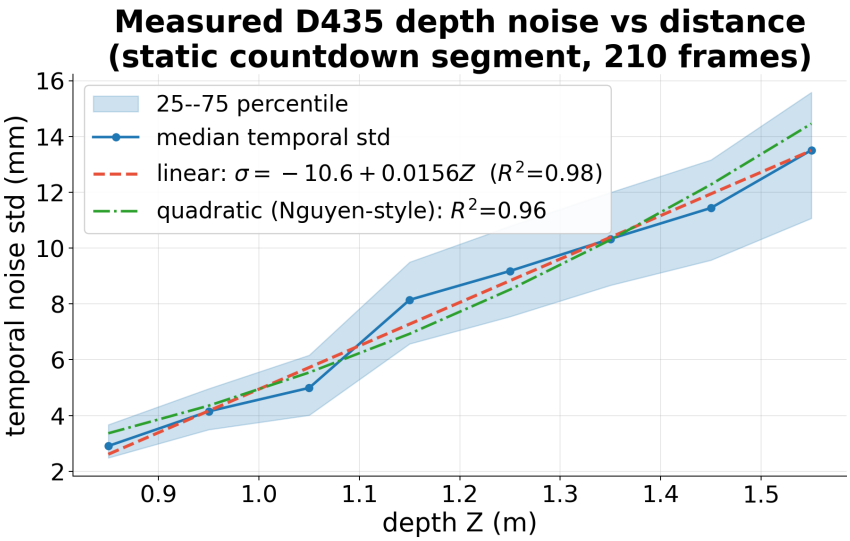


Fig. 5: Measured D435 depth noise versus distance (per-pixel temporal standard deviation over 210 static frames of my own recording), with linear and Nguyen-style quadratic fits.

pipeline are shown below for completeness; the paragraphs mark clearly which results are my own in-depth analysis and which are teammates' headline conclusions, shown here with the figure but without repeating their full derivation.

A. Noise-Model Verification

Fig. 5 is my central platform result: on our own D435, temporal depth noise rises from 3 mm at 0.85 m to 13.5 mm at 1.55 m and follows the linear law of (12) with $R^2 = 0.98$ (a Nguyen-style quadratic fits comparably, $R^2 = 0.96$). This (i) validates the assumption behind the group's synthetic-noise protocol, (ii) reproduces the qualitative finding of Nguyen *et al.* [2] on a modern active-stereo sensor, and (iii) pins the study operating point $\sigma_0 \approx 18$ mm at the top of the measured range, so every filtering experiment in the project—the 2-D study, the 3-D metric, and my own tracking comparison—stresses the filters slightly harder than the sensor's average noise. This measurement is also a control: because it uses only the static countdown segment, it cannot be confounded by the arm's own motion jitter, so I can be confident the ∼18 mm operating point reflects the sensor, not my platform's mechanics.

B. 2-D Filter Headlines (teammate B)

The full 2-D study lives in the companion report; here Fig. 6 collects only the three results that matter for my tracking comparison. **Row (a):** operating-point restoration on frame 300—median and bilateral (41.5/41.4 dB) keep box corners that mean/Gaussian blur, and bilateral has the lowest 3-D RMSE bar (6.8 mm), which is why I feed bilateral-restored depth into Experiment B. SSIM rises from 0.37 to 0.90–0.95 for every filter (Table II). **Row (b):** bilateral keeps a real depth step thin and in place (94% of reference edge strength) while median displaces it slightly (83%) and Gaussian rounds it off—sharp edges are the surface normals my point-to-plane ICP consumes in (17). **Row (c)–(d):** changing the noise *type* changes the winner outright (median under salt-and-pepper, mean under speckle, bilateral under Gaussian); hole-filling recovers 16% dropout (11.9 → 37.8 dB) for display, but inpainted pixels carry no real geometry, which is why hole-filled depth must not feed ICP below. I reuse exactly these noise types, unchanged, in Experiment B.

TABLE II: PSNR/SSIM at the D435 operating noise level ($\sigma_0 \approx 18$ mm, frame 300).

Method	PSNR (dB)	SSIM
Noisy input	31.0	0.37
Mean	40.6	**0.95**
Gaussian	40.9	0.92
Median	**41.5**	0.93
Freq. Gaussian LP	38.1	0.94
Freq. Butterworth LP	37.7	0.94
Bilateral	41.4	0.90

C. Scene Diversity (teammates B and C)

Two auxiliary scenes confirm the findings are not scene-specific: on the hand-held sofa/guitar capture the six-filter ranking repeats (median/bilateral keep the guitar neck that mean/Gaussian blur), and the hole-fill demo on my own battery frames recovers synthetic dropout while keeping the box outline. Both experiments are analyzed with their figures in the companion reports; here they matter only as evidence that the filter ranking I feed into the tracking comparison is not an artifact of one scene.

a) Fusion ablation across both scenes (teammate C).: Fig. 7 asks whether bilateral pre-filtering matters for TSDF fusion. The top row is my arm scan under amplified noise: fusing without bilateral (middle) leaves a fuzzy halo around the mesh; bilateral pre-cleaning (right) removes 12% of junk surface. The bottom row is the sofa/guitar scene at native

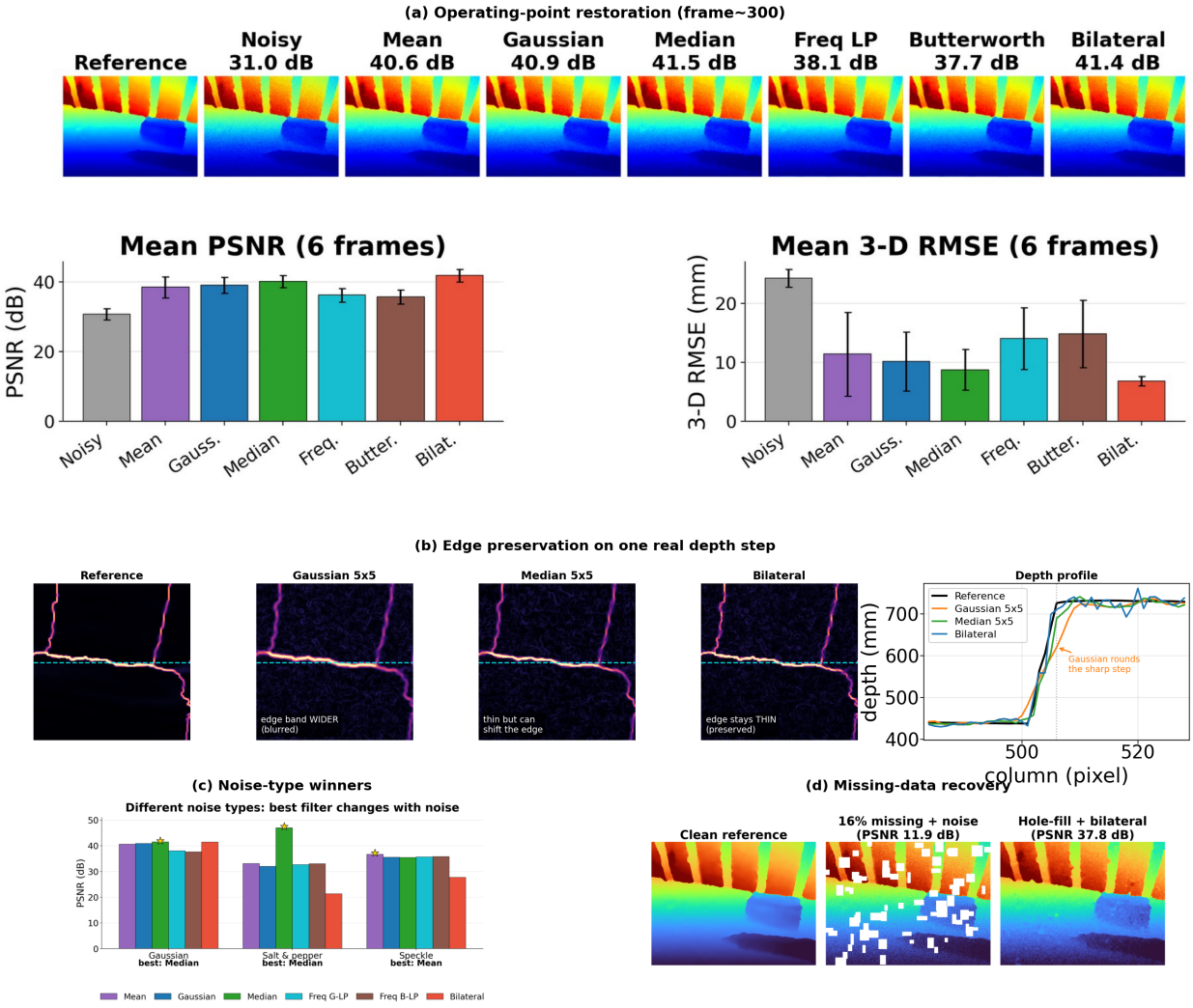


Fig. 6: 2-D filter headlines for tracking (teammate B). **(a)** Operating-point restoration and six-frame PSNR / 3-D RMSE summary. **(b)** Edge preservation on one real depth step. **(c)** Winner changes with noise type. **(d)** Missing-data recovery (hole-fill + bilateral).

noise: raw and bilateral fusion meshes are visually the same—TSDF averaging already denoises. I do not re-derive the full pipeline here; detailed 3-D analysis appears in the companion report.

D. 3-D Reconstruction (teammate C) and Closed-Loop Tracking

The pipeline (Fig. 1) fuses 300 frames of my arm sweep into a desktop mesh (∼100k vertices, 299/299 successful odometry pairs; reconstruction figures in the companion report). The battery block and desk plane are readable at millimeter scale. I do not re-derive the reconstruction implementation—that is Member C's contribution—but the mesh confirms that the scan path and odometry stack I built produces a usable geometric output.

a) Experiment A (real noise, FK reference).: On real, unamplified sensor noise (Table III), all six well-behaved filters land in a narrow ATE band of 16.7–19.0 mm against FK; frequency low-pass filters are clearly worse (37–40 mm), and aggressive hole-filling before tracking hurts (21.7 mm)—inpainted regions have no real geometry for ICP to align to.

Fig. 8 dissects *where* these numbers come from. Panel (a) overlays the Sim3-aligned odometry paths $\hat{\mathbf{p}}_k$ from Eq. (19) on my FK reference $\mathbf{p}_k^{\mathrm{FK}}$: for the best spatial-filter variant (Gaussian 5×5) the two arcs follow the same desk-height path and agree to 16.7 mm ATE (Eq. (20)); the spatial filters (blue/red) hug the reference through both taught turns, the hole-fill variant (orange) starts to cut the second corner, and the Butterworth path (brown) bends away from the arc entirely. Panel (c) shows the same sweep in 3-D: the paths stay at a nearly constant desk height ($z \approx$ constant), so the top view in (a) captures most of the motion while the 3-D panel confirms there is little vertical excursion beyond the FK calibration error. Panel (b) plots the per-frame error along the sweep and explains the ATE ordering: everyone's error dips to 5–10 mm on the slow mid-sweep segments and spikes to ∼30 mm at the two waypoint turns ($t \approx 16$ and 38 s)—those



Fig. 7: Fusion ablation across two scenes. **Top row**—Scene 1 daily-necessities desktop (arm scan, RGB frame 300): controlled noise, fused without/with bilateral pre-filtering. **Bottom row**—Scene 2 guitar and bed at native noise: raw versus bilateral fusion are nearly identical.

spikes appear for *all* variants, so they are FK-reference artifacts (interpolated waypoints, no encoders), not filter failures. What separates the variants is the final fast segment ($t > 45$ s): the frequency filter's blurred normals lose the fast wrist rotation and its error runs away to $\sim$100 mm, while hole-fill drifts to $\sim$40 mm because inpainted pixels vote in the ICP term of Eq. (15) with fabricated geometry. This per-frame view is the evidence behind my claim that the $\sim$17 mm floor is a property of the reference, not of the filters.

TABLE III: Experiment A: ATE (mm) against my calibrated FK reference, all eight pre-processing variants, real (unamplified) sensor noise.

Pre-processing	ATE (mm)	max err. (mm)
Raw (no filter)	19.0	36.8
Mean 5×5	17.3	35.2
Gaussian 5×5	**16.7**	38.2
Median 5×5	18.7	36.2
Bilateral	19.0	36.8
Freq. Gaussian LP	39.9	101.2
Freq. Butterworth LP	37.2	98.2
Hole-fill + bilateral	21.7	45.8

Analysis. The $\sim$17 mm floor here is not a filter property: it is the accuracy of my FK reference (interpolated waypoints, one-pose calibration, no encoders or hand–eye calibration). All six well-behaved filters land inside a 2.3 mm band around that floor—smaller than the floor itself—so Experiment A genuinely cannot resolve a fine-grained ranking; only the two badly-behaved variants (frequency filters, over-aggressive hole-filling) stand out clearly. This is why I designed Experiment B with an exact oracle reference instead of relying on FK alone.

b) Experiment B (controlled noise, oracle reference).: Under injected noise (Table IV), the bilateral filter drifts least under Gaussian noise (0.48 mm RPE); under salt-and-pepper it fails (11.2 mm, 30× worse than median's 0.37 mm); under speckle the median filter leads the tested set (1.51 mm) while bilateral lags (2.61 mm). This is the 2-D message—match the filter to the noise type—now confirmed at the level of camera poses rather than pixels.

TABLE IV: Experiment B: tracking drift (RPE, mm) against the oracle trajectory. Best per column in bold.

Filter	Gaussian	Salt & pepper	Speckle
None (noisy)	1.30	12.7	4.06
Gaussian	0.50	3.00	1.73
Median	0.50	**0.37**	**1.51**
Bilateral	**0.48**	11.2	2.61
Freq. Butterworth	2.69	2.08	2.19

Fig. 9 visualizes both experiments side by side: panel (a) shows the narrow ATE band of Experiment A (all spatial filters within $\sim$2 mm of each other, frequency filters and hole-fill standing out); panel (b) shows Experiment B's filter–noise match, with stars marking the winner per noise type. *Analysis.* Because Experiment B's oracle is exact by construction, it—not Experiment A—carries the group's tracking conclusion; the FK experiment should be read as "all reasonable spatial filters are equivalent at native noise," not as a fine-grained ranking. A kinematic reference is good enough to catch large errors but not to resolve differences at the few-millimeter level that an oracle reveals.

E. Consistency with the Original Works

My results agree with both source papers and add boundary conditions they do not report. KinectFusion's bilateral choice [1] is confirmed in its intended regime—best edges, best low-noise PSNR, best Gaussian-noise tracking—and Nguyen's message [2] (model and respect the sensor noise) appears twice in my own measurements: our sensor obeys the distance-dependent law ($R^2 = 0.98$), and mismatching filter to noise type costs an order of magnitude in tracking drift. The new boundary condition from my platform: on real, unamplified noise, the spatial-filter choice barely affects

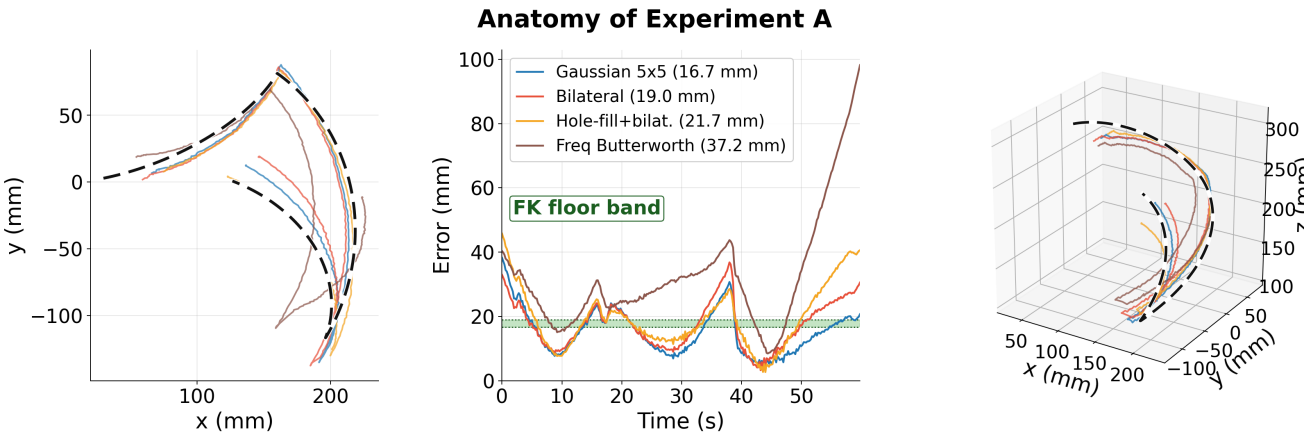
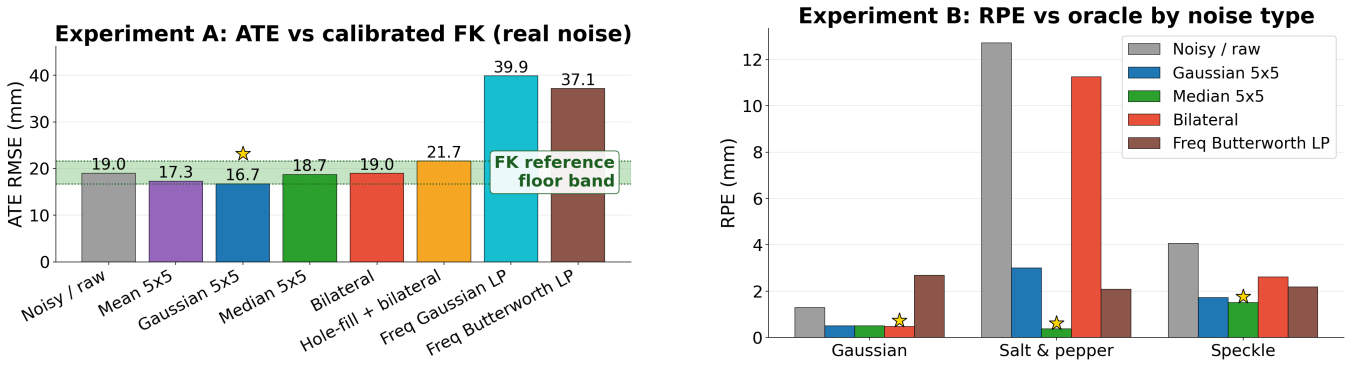


Fig. 8: Anatomy of Experiment A.



(a) Experiment A: ATE vs FK (real sensor noise).

(b) Experiment B: RPE vs oracle by noise type.

Fig. 9: Closed-loop tracking comparison (Experiments A and B).

tracking at all, because the FK reference's own accuracy dominates the error budget.

V. Discussion and Limitations

The practical recipe from the project is short: use bilateral by default; switch to median when impulse-like artifacts appear; never inpaint holes before tracking. The main limitation is one I own: FK bounds rather than measures tracking error, since I have no joint encoders or hand–eye calibration, so Experiment B's oracle carries the tracking conclusion rather than Experiment A. A second limitation, shared with the group, is that the clean reference used throughout is itself built with a mild bilateral pass, which may slightly flatter bilateral's absolute edge score; the ranking is still safe because median and mean win their respective noise types outright.

VI. Conclusion and Future Work

The group replicated the classical depth-restoration evaluation on real RealSense D435 data and extended it with a 3-D point-distance metric, multiple noise conditions, a scene-diversity check, and a closed-loop reconstruction-and-tracking validation on the robot-arm platform I built, whose scan path was planned in task space, executed through IK, and referenced through MuJoCo-calibrated forward kinematics. There is no universal best filter: the bilateral filter is the right default at the sensor's operating point, while the median filter is decisively more robust to impulse and extreme noise, and my tracking comparison shows this same ranking holds all the way to camera poses. **My own future work**: joint encoders or motion-capture ground truth with hand–eye calibration, to turn my FK bound into a true ground truth capable of resolving the fine-grained ranking that Experiment A currently cannot.

Member Contributions

The work was split vertically: each member ran one complete study line from literature review through methodology, experiments, analysis, and conclusions.

Member A — Chuhan ZHANG (robot-arm platform, sensor analysis, and the final tracking comparison, me). Literature on active-stereo sensing and noise models [2], [10]; hardware setup and wiring; task-space scan planning, IK execution, MuJoCo FK calibration; capture of the 1798-frame dataset; the static-segment noise measurement; and the closed-loop tracking comparison (Experiments A and B). Conclusions: the noise model of (12) holds ($R^2 = 0.98$) and fixes the operating point; the filter–noise match decides tracking drift; the $\sim$17 mm floor of Experiment A comes from the FK reference itself.

Member B — Jianting WU (2-D restoration study). Literature on classical filters and metrics [3]–[5]; implementation of the six filters and the noise–restore–score protocol; the operating-point, edge, noise-level, noise-type, missing-data, and scene-diversity experiments. Conclusions: bilateral is the best default; median wins under impulse noise; no filter wins everywhere; 16% missing data is recoverable by hole-filling.

Member C — Junzhe XU (3-D geometry and reconstruction). Literature on KinectFusion and learning-based restoration [1], [6], [7]; the pinhole back-projection metric; the ICP + TSDF pipeline; the 3-D RMSE evaluation; and the final TSDF point-cloud/mesh reconstructions of both scenes. Conclusions: restoration cuts 3-D geometric error from 26.0 to 6.8–7.8 mm; fusing raw depth keeps detail while filtered depth feeds tracking, reproducing KinectFusion's design.

REFERENCES

[1] R. A. Newcombe *et al.*, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. IEEE ISMAR*, 2011, pp. 127–136.

[2] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling Kinect sensor noise for improved 3D reconstruction and tracking," in *Proc. 3DIMPVT*, 2012, pp. 524–530.

[3] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. IEEE ICCV*, 1998, pp. 839–846.

[4] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

[5] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. Pearson, 2018.

[6] V. Sterzentsenko *et al.*, "Self-supervised deep depth denoising," in *Proc. IEEE ICCV*, 2019, pp. 1242–1251.

[7] A. Duarte *et al.*, "SelfReDepth: Self-supervised real-time depth restoration for consumer-grade RGB-D sensors," arXiv:2406.03388, 2024.

[8] S. Wei *et al.*, "D3RoMa: Disparity diffusion-based depth sensing for material-agnostic robotic manipulation," arXiv:2409.14365, 2024.

[9] T. Sun, D. Hu, Y. Dai, and G. Wang, "Diffusion-based depth inpainting for transparent and reflective objects," *IEEE Trans. Circuits Syst. Video Technol.*, 2024.

[10] F. Afzal Maken *et al.*, "Improving 3D reconstruction through RGB-D sensor noise modeling," *Sensors*, vol. 25, no. 3, art. 950, 2025.

[11] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 4, pp. 376–380, 1991.

[12] *Digital Image and Signal Processing*, Course 236327 Lecture Notes and Tutorials (Lectures 2 and 6; Tutorials 7–8), GTIIT Mechanical Engineering Program, 2026.