Vision-Guided Autonomous Wound Scanning and Treatment System

IE Machine Vision 549 - Final Project Report

Daniel Jacquet Praveen Krisna

Purdue University

April 30, 2026

Contribution Statement

This project was built through close collaboration. Both authors participated in design decisions, integration, and evaluation together. The division below reflects each person's primary ownership area.

Daniel Jacquet led the robotics and systems engineering side: ROS2 architecture design, SVD-based surface normal estimation, kinematically-filtered arc viewpoint generation, the Gazebo simulation scene, and the locking mechanism (FSM design, median-filter orchestration, Pilz PTP integration).

Praveen Krisna led the machine vision side: wound image dataset curation and manual annotation, YOLO model training and evaluation, wound report collector implementation, anatomical localization via skeleton-to-segment projection, and colorimetric severity analysis.

Both authors jointly developed the skeleton state-freezing strategy, 3D spatial deduplication, full system integration, and the evaluation protocol. The core perception-actuation loop reflects continuous back-and-forth between both sides, and neither pipeline could work without the other's contributions.

Contents

1 Abstract **4**

2 Introduction **4**

2.1 Motivation 4

2.2 Problem Definition 4

2.3 Key Challenges 5

2.4 Summary of Contributions 5

3 Related Work **5**

3.1 Wound Detection and Segmentation 5

3.2 Robot-Assisted Surface Treatment 6

3.3 Positioning of Our Approach 6

4 Methodology **6**

4.1 System Overview 6

4.1.1 ROS2 Node Graph 7

4.2 Hardware Setup 8

4.3 Dataset 8

4.4 System Calibration 9

4.4.1 Camera Intrinsic Calibration 9

4.4.2 Depth-to-RGB Registration 9

4.4.3 Eye-in-Hand Calibration 10

4.5 Pipeline 1: Autonomous Scanning (Physical Robot) 10

4.5.1 Anatomical Tracking and Skeleton Freeze 10

4.5.2 Wound Detection 10

4.5.3 3D Surface Normal Estimation via SVD 10

4.5.4 Roll-Minimized Target Frame 11

4.5.5 Viewpoint Generation and Kinematic Filtering 11

4.5.6 3D Spatial Deduplication 12

4.5.7 Anatomical Localization 12

4.5.8 Colorimetric Severity Index 13

4.6 Pipeline 2: Real-Time Locking Mechanism (Simulation) 14
4.6.1 Visual Feature Detection 14
4.6.2 3D Pose and Standoff Calculation 14
4.6.3 FSM Orchestration and Median Filtering 14
4.6.4 Active Locking and Micro-Corrections 15
4.6.5 Integration Path to the Physical Robot 15
4.7 YOLO Training Details 15

5 Experiments **16**
5.1 Experimental Setup 16
5.2 Evaluation Metrics 16
5.3 Baselines 16

6 Results **17**
6.1 YOLO Model Performance 17
6.2 Full-System Scanning Performance 17
6.3 Locking Mechanism Qualitative Results (Simulation) 18
6.4 Discussion 19

7 Conclusion **19**

1. Abstract

Wound assessment in clinical and hazardous environments is typically manual, inconsistent, and difficult to scale. This paper presents a two-pipeline robotic system built on a Universal Robots UR16e manipulator that automates wound localization, surface characterization, and treatment alignment using computer vision and ROS2.

Pipeline 1 runs on the physical robot. A Kinect V2 RGB-D sensor mounted eye-in-hand streams data through a ROS2 node graph that detects wounds with a fine-tuned YOLO model, estimates their 3D surface normals via Singular Value Decomposition (SVD), plans and executes a three-pose arc scan guided by a frozen MediaPipe skeleton, deduplicates detections in world space, and produces a per-wound clinical report.

Pipeline 2 is validated in Gazebo simulation. A real-time locking mechanism uses HSV-based detection, SVD pose estimation, and a Finite State Machine (FSM) to bring the UR16e to a perpendicular 15 cm standoff and actively maintain alignment through patient motion via threshold-gated Pilz PTP micro-corrections. The pipeline is architecturally isolated and designed for straightforward deployment onto the physical system.

The fine-tuned YOLO model achieves mAP@50 of 73% (precision 0.88, recall 0.63) on a 140-image custom dataset. Full-system evaluation across four scanning sessions of 14 wounds each yields a detection rate of 71.43% and anatomical localization accuracy of 75.62%.

2. Introduction

2.1 Motivation

Manual wound assessment relies on a clinician visually inspecting a wound, estimating its size and severity, and tracking changes over time. This process is inherently subjective, difficult to reproduce across practitioners, and time-intensive, particularly in high-volume or resource-limited settings. Beyond clinical care, wound treatment in hazardous environments (radiation zones, contaminated sites, disaster response) creates direct risk to human operators. Since the Chernobyl disaster, deploying robotic systems to minimize human exposure in such conditions has been an active research focus [8]. An autonomous, reproducible wound scanning and treatment system addresses both contexts simultaneously.

2.2 Problem Definition

We define the task as two coupled problems.

Scanning: Given a synchronized RGB-D stream from a robot-mounted camera, a patient body model estimated from a single standoff view, and a 6-DOF manipulator, autonomously detect all visible wounds on the patient's surface, estimate each wound's 3D centroid and surface normal, and produce a structured clinical report including anatomical location and colorimetric severity.

Treatment locking: Given a real-time RGB-D stream and the robot's current end-effector pose, maintain a perpendicular 15 cm standoff to the wound surface and continuously correct for patient motion without spamming the motion planner in a way that degrades performance.

2.3 Key Challenges

Several engineering challenges had to be addressed across this project:

- **Kinect V2 minimum depth range** (≈50 cm): prevents close-up surface scanning at treatment standoff distances.
- **FOV clipping during close-up scans**: as the robot moves in, the camera loses sight of the patient's full body, breaking skeleton tracking. Solved by keeping a memory of the 3D skeleton from the initial standoff pose.
- **IK feasibility along an arc sweep**: candidate viewpoints must be reachable, singularity-free, and connected by a safe continuous joint path.
- **Duplicate detections across scan poses**: the same wound seen from Left, Center, and Right must be collapsed into one report entry.
- **Real-time locking stability**: raw frame-by-frame visual commands cause jitter that is unsafe and mechanically stressful.

2.4 Summary of Contributions

- Custom fine-tuned YOLO wound detector trained on a 140-image dataset.
- Complete eye-in-hand calibration pipeline linking camera pixels to robot world coordinates.
- SVD-based 3D surface normal estimation with roll-minimized target frame generation.
- Kinematically-filtered arc viewpoint generation with wrist-travel path validation.
- 3D spatial deduplication and skeleton-anchored anatomical localization.
- FSM-based real-time locking mechanism with median-filtered pose smoothing and Pilz PTP micro-corrections.

3. Related Work

3.1 Wound Detection and Segmentation

Deep learning has proven effective for automated wound analysis. Wang et al. [1] demonstrated fully automatic wound segmentation with convolutional neural networks, establishing that deep models can produce clinically useful wound boundaries. Aldughayfiq et al. [2] applied YOLO-based detection to pressure ulcer classification, showing that single-stage detectors run fast enough for real-time screening. Jacob et al. [3] extended YOLO-based wound detection to multi-ethnic populations, highlighting the generalization challenges that motivate domain-specific fine-tuning. Velázquez et al. [4] used YOLOv7 for wound size measurement, the

closest prior work to ours in chaining YOLO bounding boxes into a downstream processing pipeline.

Our contribution extends this body of work by connecting YOLO detections to 3D surface estimation and robot actuation, rather than treating detection as a standalone output.

3.2 Robot-Assisted Surface Treatment

Thakar et al. [5] developed area-coverage planning for spray-based surface disinfection with a mobile manipulator, demonstrating that surface geometry can drive robotic treatment paths. Nieto Bastida and Lin [6] applied autonomous trajectory planning to spray painting on complex 3D surfaces from point cloud models, directly relevant to our surface-normal-aligned treatment geometry. Vargas et al. [7] coupled deep learning wound detection with finite element modeling to drive a robotic suturing system, demonstrating the feasibility of perception-driven autonomous wound intervention. Lee et al. [8] survey autonomy levels in FDA-cleared surgical robots, providing context for where our system sits relative to the state of clinical deployment.

3.3 Positioning of Our Approach

Existing systems focus either on detection without actuation, or on actuation without closed-loop surface-normal alignment. Our system is self-contained (eye-in-hand, no external tracking infrastructure), generates multi-view coverage through active robot repositioning, and maintains perpendicular alignment dynamically. The skeleton state-freezing strategy and threshold-gated Pilz micro-corrections are design patterns not present in prior literature.

4. Methodology

4.1 System Overview

The system is built entirely on ROS2 and MoveIt2, interfacing with a UR16e manipulator. It comprises two modular, independently launchable pipelines that share the same hardware interface and topic naming convention, as shown in Figure 1.

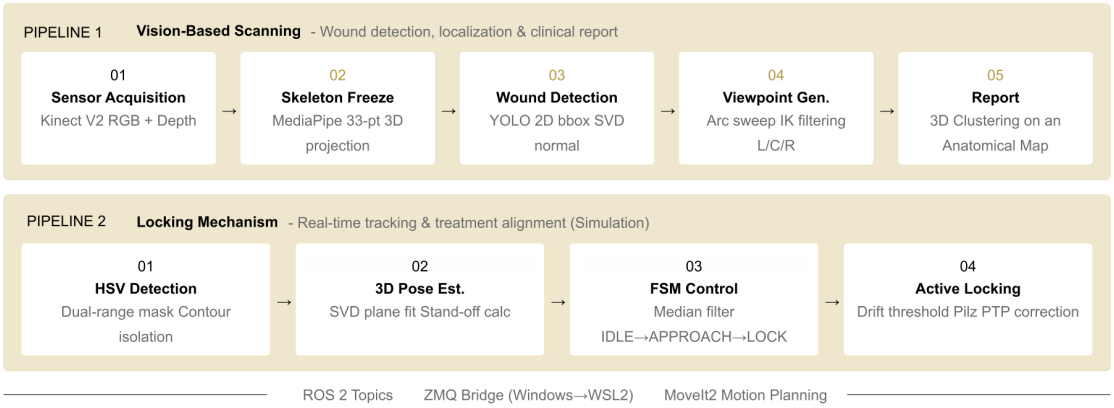

Figure 1: Two-pipeline system architecture. Pipeline 1 (Scanning) runs on the physical robot. Pipeline 2 (Locking) runs in Gazebo simulation and is designed for drop-in deployment on the physical system.

4.1.1 ROS2 Node Graph

Tables 1 and 2 list the ROS2 nodes for each pipeline. Both pipelines publish and subscribe to the same camera topic names (`/camera/color/image_raw`, `/camera/depth_registered/image_raw`). The `camera_bridge` node in Pipeline 2 remaps Gazebo's internal camera topics to these names, meaning all downstream nodes are identical to their physical counterparts.

Table 1: Pipeline 1 ROS2 nodes (physical scanning system, `full_pipeline.launch.py`).

Node	**Publishes**	**Subscribes**
`kinect2_driver`	`/camera/color/image_raw`, `/camera/depth_registered/image_raw`, `/camera/color/camera_info`	–
`calibrated_tf_publisher`	TF: `tool0`→ `camera_color_optical_frame`	–
`skeleton_tracker`	`/skeleton/landmarks`	RGB + depth
`scan_pose_arc`	`/scan_arc/joint_targets`	`/skeleton/landmarks`
`scan_loop`	`/scan/at_pose`	joint targets, joint states
`yolo_pose`	`/wound/pose`, `/wound/bbox`	RGB + depth + TF
`wound_report_collector`	`/wound/report`	`/scan/at_pose`, `/wound/pose`

Table 2: Pipeline 2 ROS2 nodes (simulation locking system, `sim_mannequin.launch.py`).

Node	Role
`ur_sim_control` (Gazebo)	Physics simulation + UR16e ros2_control
`ur_moveit`	MoveIt2 motion planning + RViz
`camera_bridge`	Remaps `/wound_camera/*` → `/camera/*` (same names as Pipeline 1)
`feature_detector`	HSV dual-range wound ROI detection
`pose_estimation`	SVD plane fit, publishes 3D wound pose
`orchestrator`	FSM control + median-filter pose smoothing
`approach_node`	Macro MoveIt2 movement to standoff pose
`locking_node`	Drift monitoring + Pilz PTP micro-corrections

4.2 Hardware Setup

Figure 2: Hardware setup displaying the UR16e (left), the Kinect V2 sensor (middle), and a Fusion 360 visualization of the custom mount integrating both components (right).

The UR16e is a 6-DOF collaborative manipulator with 900 mm reach, 16 kg payload, and ±0.05 mm repeatability, controlled via MoveIt2 [15] using the `ur_ros2_driver`. The Kinect V2 captures RGB at 1920×1080 @ 30 fps and depth at 512×424 via Time-of-Flight, with a hardware RGB-Depth registration capability and a minimum operating depth of approximately 50 cm. The camera is rigidly attached to the robot's `tool0` frame via a custom 3D-printed bracket, forming the eye-in-hand configuration. A ZMQ bridge handles streaming from the Kinect's Windows SDK to the ROS2 stack [14] running on Ubuntu/WSL2, bypassing Windows-to-Linux networking bottlenecks.

4.3 Dataset

We built a custom wound image dataset for YOLO fine-tuning. The 140 images were sourced from the Medetec Wound Database [9], supplemented with additional wound photography covering diverse tissue types (lacerations, abrasions, pressure ulcers) and skin tones. Each

image was manually annotated with bounding boxes in YOLO label format. The dataset was split approximately 80/10/10 into training, validation, and test sets, and all images were resized to 640×640 to fit GPU memory constraints. The Ultralytics YOLOv8 default augmentation suite (mosaic, random flips, HSV jitter, random scale) was applied during training to compensate for the small corpus size. At 140 images, this is an intentionally minimal dataset, sufficient to demonstrate domain-specific fine-tuning, and its size directly bounds the model's recall.

4.4 System Calibration

Before either pipeline runs, four calibration steps establish the geometric links that make accurate 3D wound localization possible.

4.4.1 Camera Intrinsic Calibration

A hardware change mid-project, from an initially planned uncalibrated RGB-D camera to the Kinect v2, introduced a calibration inconsistency that shaped the final approach. The Kinect v2 stores per-unit factory intrinsics for both its RGB and depth sensors directly on the device; the Kinect for Windows SDK exposes these through `CoordinateMapper` and `GetDepthCameraIntrinsics`, meaning depth registration does not require a separate calibration step. However, because the project had already committed to an OpenCV calibration workflow designed for the original uncalibrated camera, an explicit RGB intrinsic calibration was retained and ultimately proved complementary: the SDK's `MapColorFrameToDepthSpace` handles depth-to-color alignment using factory parameters internally, while the OpenCV-derived intrinsic matrix K and distortion vector $\mathbf{d}$ are published via the ROS2 `CameraInfo` topic and consumed by downstream 3D deprojection nodes. Calibration used a 9×6 checkerboard imaged from diverse orientations; `findChessboardCorners` with `cornerSubPix` refinement extracts corners at subpixel accuracy, and `calibrateCamera` solves for K and $\mathbf{d}$ by minimizing reprojection error.

4.4.2 Depth-to-RGB Registration

The Kinect's RGB and depth sensors are physically offset, so a pixel in the RGB frame does not point to the same physical location as the same pixel in the raw depth frame. We use the Kinect SDK's `MapColorFrameToDepthSpace` to warp the 512×424 depth frame into the 1920×1080 RGB coordinate space, leveraging the per-unit factory extrinsics stored on the device. When downscaling to the pipeline's operating resolution (960×540), nearest-neighbor interpolation is strictly enforced, bilinear averaging would blend raw `uint16` depth values at object boundaries, introducing geometrically corrupt edge pixels before SVD.

4.4.3 Eye-in-Hand Calibration

The camera moves with the robot's wrist, so its world pose changes with every joint configuration. Eye-in-hand calibration computes the static transform $T_{cam \to tool0}$ between the camera's optical frame and the robot's `tool0` frame. Once known, this transform is constant regardless of arm configuration, and the full TF chain $T^{base}_{wound} = T^{base}_{tool0} \cdot T^{tool0}_{cam} \cdot T^{cam}_{wound}$ places any camera-frame detection into world coordinates.

We solve the standard $AX = XB$ problem: the robot is moved through 20+ poses with high rotational diversity while a static checkerboard is in view. For each pose, the robot reports its gripper-to-base transform and the vision system computes the checkerboard-to-camera transform. OpenCV's `calibrateHandEye` with Tsai's method [13] yields the calibration rotation matrix and translation vector. A dedicated ROS2 node loads these from a JSON file at startup, converts to a quaternion, and broadcasts the static TF continuously.

4.5 Pipeline 1: Autonomous Scanning (Physical Robot)

4.5.1 Anatomical Tracking and Skeleton Freeze

Wound localization requires knowing which body part a wound is near. We use MediaPipe Pose [12] to detect 33 anatomical landmarks in the RGB frame. Each 2D landmark (u, v) is projected into 3D by reading the registered depth value Z at that pixel (converted mm $\to$ m) and applying the pinhole model:

$$X = \frac{(u - c_x) \cdot Z}{f_x}, \qquad Y = \frac{(v - c_y) \cdot Z}{f_y} \tag{1}$$

From the initial standoff pose, where the full body is visible, we detect a complete 33-point skeleton, transform all 3D landmarks into the static `base_link` frame, and freeze them. This is the key design choice that makes the rest of the pipeline possible: as the robot moves into close-up scan poses and the Kinect's field of view clips the patient's body, the frozen skeleton remains a stable reference for both path planning and wound localization.

4.5.2 Wound Detection

A YOLOv8 model [11] fine-tuned on our custom dataset processes each RGB frame to produce 2D bounding boxes $(x_{min}, y_{min}, x_{max}, y_{max})$ with confidence scores. The model runs at each of the three scan poses; all detections feed the deduplication step described in Section 4.5.6.

4.5.3 3D Surface Normal Estimation via SVD

Knowing where a wound is in 2D is not sufficient for robot alignment, the robot also needs the wound's 3D orientation to approach perpendicular to the skin. For each YOLO bounding box, all internal pixels are deprojected into 3D using Equation (1) and the registered depth.

Depth pixels deviating more than 2σ from the patch median are discarded as noise before fitting.

The remaining $N \times 3$ point cloud M is mean-centered and decomposed via SVD:

$$M - \bar{M} = U \, \Sigma \, V^T \tag{2}$$

The surface normal $\mathbf{n}$ corresponds to the last row of V^T, the direction of minimum variance, orthogonal to the best-fitting plane. If $n_z > 0$ (normal pointing away from the camera), it is negated so it always points toward the camera. This normal directly defines the robot's required approach direction.

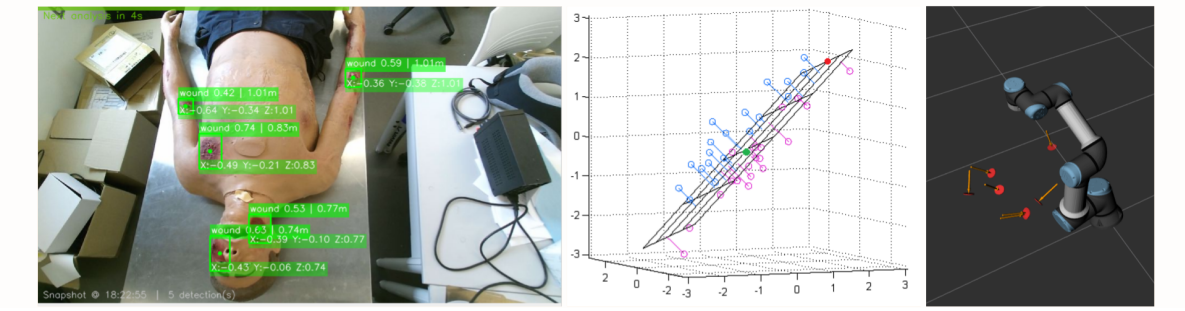

Figure 3: Wound detection and surface normal estimation. YOLO detects the wound in 2D (left). The corresponding depth patch is deprojected into 3D; SVD fits a plane and extracts the surface normal used for perpendicular robot alignment (right).

4.5.4 Roll-Minimized Target Frame

With the surface normal as the new approach Z-axis, the robot's wrist roll is unconstrained, any rotation around Z is geometrically valid for an axially symmetric tool. Without constraining roll, the wrist can spin arbitrarily between frames as the mathematically arbitrary component changes. We resolve this by projecting the current end-effector X-axis onto the new target plane:

$$\mathbf{x}_{new} = \frac{\mathbf{x}_{cur} - (\mathbf{x}_{cur} \cdot \mathbf{n})\,\mathbf{n}}{\|\mathbf{x}_{cur} - (\mathbf{x}_{cur} \cdot \mathbf{n})\,\mathbf{n}\|} \tag{3}$$

$$\mathbf{y}_{new} = \mathbf{n} \times \mathbf{x}_{new} \tag{4}$$

This minimizes wrist travel between consecutive target updates, producing smooth and mechanically safe motion.

4.5.5 Viewpoint Generation and Kinematic Filtering

A single frontal view rarely captures all wounds on a patient's body. We generate three scan viewpoints (Left, Center, Right) by sweeping a camera arc around the patient's torso, as illustrated in Figure 4.

The torso center C is computed by averaging the frozen 3D positions of the shoulder and hip landmarks. A candidate camera position $\mathbf{p}(\theta)$ is placed at a fixed radius from C, parameterized by angle θ. For each candidate, the optical Z-axis is aimed toward C:

$$\mathbf{z}_{opt}(\theta) = \frac{C - \mathbf{p}(\theta)}{\|C - \mathbf{p}(\theta)\|} \tag{5}$$

Each candidate pose is passed through the UR16e IK solver. Solutions are scored with a manipulability proxy that penalizes straight-arm and locked-wrist configurations:

$$m = |\sin(q_4)| \cdot |\sin(q_2)| \tag{6}$$

where q_2 and q_4 are the elbow and wrist joint angles. Only solutions above a minimum manipulability threshold are accepted. The final Left, Center, and Right poses are additionally validated jointly: the linear joint-space path between each consecutive pair is simulated, and wrist rotation is capped at $|\Delta q_6| < 2.5$ rad between poses, ensuring a safe, continuous execution sequence.

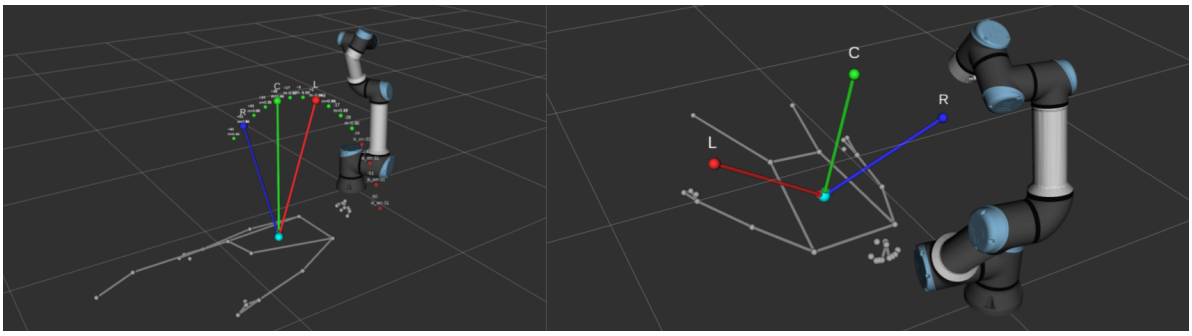

Figure 4: Arc viewpoint generation. Torso center C from the detected skeleton. Angle θ parameterizes camera position along the arc. IK filtering and manipulability scoring select three final scan poses.

4.5.6 3D Spatial Deduplication

Scanning from three angles means the same wound appears in multiple frames. Each new YOLO detection is transformed from camera frame to `base_link` via the live TF chain. Its 3D centroid is compared against all previously stored wound centroids. If the Euclidean distance is below 0.08 m, the detection is discarded as a duplicate; otherwise it creates a new entry. This threshold was chosen empirically to be large enough to absorb cross-pose registration noise while remaining small enough not to merge nearby distinct wounds.

4.5.7 Anatomical Localization

To label each wound (e.g., "left forearm," "right chest"), we compute the shortest distance from the wound centroid $\mathbf{w}$ to each bone segment of the frozen skeleton, defined by endpoint

pairs $(\mathbf{a}, \mathbf{b})$:

$$t = \text{clamp}\left(\frac{(\mathbf{w}-\mathbf{a})\cdot(\mathbf{b}-\mathbf{a})}{\|\mathbf{b}-\mathbf{a}\|^2},\ 0,\ 1\right), \qquad d = \|\mathbf{w} - (\mathbf{a} + t(\mathbf{b}-\mathbf{a}))\| \tag{7}$$

The wound is assigned to the segment with minimum d, covering all 32 bone segments of the MediaPipe skeleton.

4.5.8 Colorimetric Severity Index

A redness index is computed from the wound's RGB crop as a proxy for inflammation or active bleeding:

$$R_{index} = \frac{\mu_R}{\mu_G + \mu_B + 1} \tag{8}$$

where μ_R, μ_G, μ_B are the mean per-channel values in the bounding box region. The final clinical report lists per wound: 3D location in `base_link`, anatomical label, R_{index}, and detection confidence.

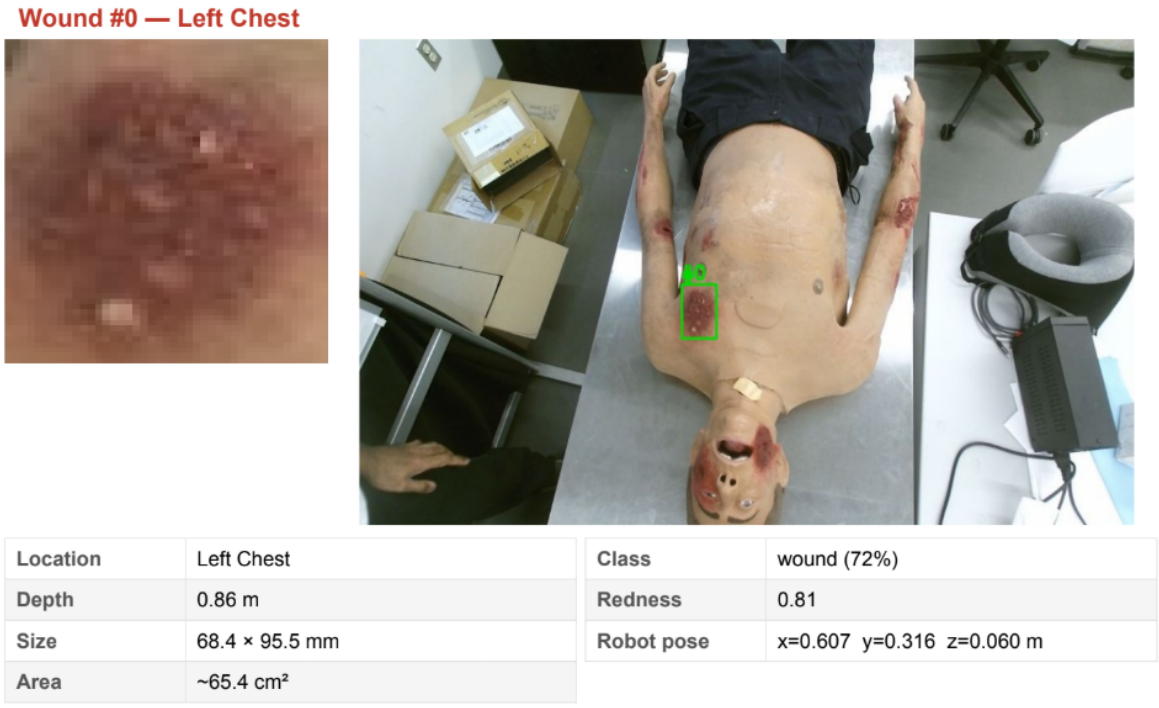

Figure 5: Clinical report generation. Each unique wound entry includes its 3D world position, anatomical label from skeleton projection, redness severity index, and YOLO detection confidence.

4.6 Pipeline 2: Real-Time Locking Mechanism (Simulation)

4.6.1 Visual Feature Detection

The locking pipeline uses HSV-based detection rather than YOLO. In the treatment phase, the wound location is already known from scanning; the goal is high-frequency, lightweight tracking, not classification. Converting the frame to HSV and applying a dual-range mask handles the hue wrapping at the 0°/180° boundary that red occupies:

$$\text{mask} = \text{inRange}(L_1, H_1) \mid \text{inRange}(L_2, H_2) \tag{9}$$

Morphological open and close operations clean noise. The largest contour is taken as the wound ROI. This approach runs at camera frame rate without any GPU requirement, which matters for a control-rate feedback loop.

4.6.2 3D Pose and Standoff Calculation

The same SVD plane-fitting procedure from Pipeline 1 is applied to the depth patch inside the detected ROI. The treatment target is placed 15 cm along the negative surface normal:

$$\mathbf{p}_{target} = \mathbf{c}_{wound} - 0.15\,\mathbf{n} \tag{10}$$

Roll minimization (Equations 3–4) is applied identically, preventing wrist instability during continuous tracking.

4.6.3 FSM Orchestration and Median Filtering

Raw frame-by-frame 3D estimates are too noisy to send directly to the motion planner. A sliding deque buffer accumulates recent pose estimates; the median spatial position and a SLERP-averaged quaternion produce a smoothed target that rejects transient outliers without introducing large lag.

An FSM manages the pipeline's behavior across three states, illustrated in Figure 6:

- **IDLE**: Buffers incoming poses until a stable, low-variance target is established. Fires the APPROACH trigger once confidence is sufficient.
- **APPROACH**: Issues a full MoveIt2 plan-and-execute request to move the end-effector to the standoff pose. Waits for confirmation before transitioning.
- **LOCKING**: Shrinks the sliding window for faster response. Continuously monitors drift. Reverts to IDLE on wound detection timeout.

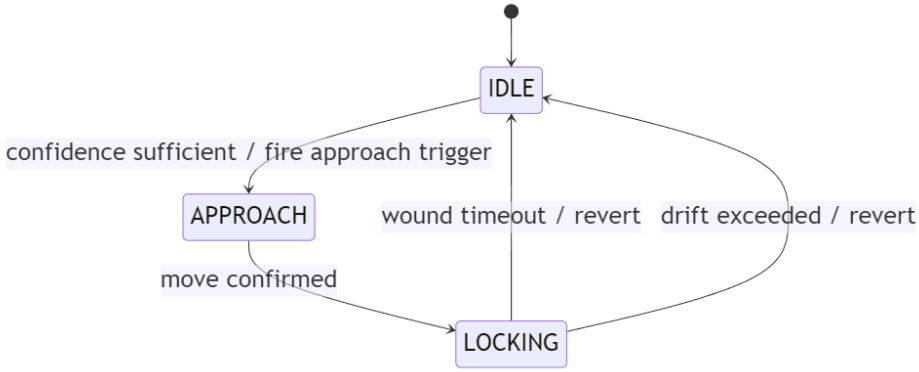

Figure 6: Locking FSM. IDLE buffers and stabilizes. APPROACH executes the macro movement. LOCKING tracks and applies threshold-gated Pilz PTP micro-corrections.

4.6.4 Active Locking and Micro-Corrections

In LOCKING state, the system continuously computes positional drift (Euclidean distance) and orientational drift (quaternion angular distance) between the current end-effector and the smoothed target. A correction is issued only when:

$$d_{pos} > 1.5\,\text{cm} \quad \text{or} \quad d_{ori} > 8.6^{\circ} \tag{11}$$

Corrections use the **Pilz Industrial Motion Planner** in PTP mode. Pilz PTP generates velocity-bounded, time-optimal point-to-point trajectories deterministically, without the overhead of OMPL-based planners. This makes corrections fast and bounded. The threshold-gating prevents the planner from being flooded by sensor noise while remaining responsive to genuine patient motion such as breathing or repositioning.

4.6.5 Integration Path to the Physical Robot

The locking mechanism is a self-contained ROS2 package. In `sim_mannequin.launch.py`, the Gazebo simulation and `camera_bridge` node supply camera topics under exactly the same names that `kinect2_driver` uses in `full_pipeline.launch.py`. Deploying the locking mechanism on the physical UR16e therefore requires only removing the Gazebo and bridge nodes and substituting `kinect2_driver` and `calibrated_tf_publisher`. Every locking node runs without any code modification. This was a deliberate design decision: keeping the simulation and physical interfaces identical reduces integration risk and makes the path to physical deployment direct.

4.7 YOLO Training Details

We fine-tuned a YOLOv8 model [11] initialized from COCO pretrained weights using the Ultralytics framework. Training ran for 500 epochs with SGD, batch size 4, and 640×640

input resolution, constrained by available GPU VRAM. The small batch size was partially compensated by mosaic augmentation, which combines four training images into one, effectively quadrupling the visual diversity seen per parameter update. Training was performed on an NVIDIA RTX 5070 Ti.

5. Experiments

5.1 Experimental Setup

Physical scanning (Pipeline 1). A full-body mannequin was placed on a table in front of the UR16e. Red adhesive wound patches of consistent size were applied at known 3D positions across the mannequin's body, covering the torso, arms, and legs. The robot executed the full scanning sequence: initial standoff to freeze the skeleton, then Left, Center, and Right scan poses while collecting YOLO detections. The generated report was compared against the known ground-truth wound count and positions.

Simulation locking (Pipeline 2). The Gazebo scene (`tracking_scene_mannequin.sdf`) places the UR16e base at $z = 0.75$ m and a humanoid mannequin on a table with a red wound patch at world coordinates (0.53, 0.0, 0.895) m. The locking pipeline was run with and without manual displacement of the wound patch to evaluate FSM state transitions and micro-correction behavior.

5.2 Evaluation Metrics

- **YOLO**: mAP@50, precision, and recall on the held-out test set.
- **Detection rate**: fraction of placed wounds detected at least once across the three scan poses.
- **Localization accuracy**: fraction of detected wounds assigned to the correct anatomical segment.
- **Locking (qualitative)**: FSM state progression correctness, steady-state stability, and responsiveness to wound displacement.

5.3 Baselines

We compare the three-pose arc scan against a single-pose (Center only) evaluation to quantify the detection gain from active repositioning. We also compare pre-trained YOLO (no fine-tuning) against the fine-tuned model to justify the domain-specific training.

6. Results

6.1 YOLO Model Performance

Table 3 summarizes the fine-tuned model's performance. Precision of 0.88 means that when the model predicts a wound, it is correct 88% of the time. Recall of 0.63 means approximately one third of wounds are missed. A pre-trained YOLO model without domain fine-tuning produced near-zero detections on wound imagery, confirming that the custom training step is necessary. Figure 7 shows validation predictions.

Table 3: Fine-tuned YOLO model performance on the held-out test set.

Metric	Value
mAP@50	73%
Precision	0.88
Recall	0.63
Training images	140
Training epochs	500

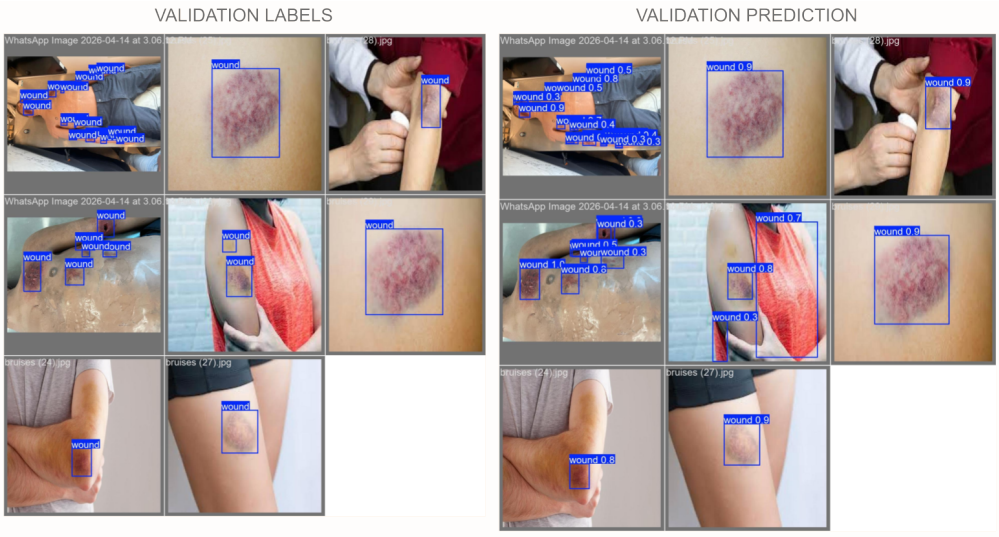

Figure 7: YOLO validation results. Ground-truth labels (left) vs. model predictions (right). The model is conservative, minimizing false positives at the cost of some missed detections.

6.2 Full-System Scanning Performance

Table 4 shows session-by-session results across four evaluations, each with 14 wound patches on the mannequin. The three-pose arc scan achieves 71.43% overall detection. In the single-pose (Center only) baseline, a representative session detected 9 of 14 wounds (64.3%), confirming that the multi-pose strategy provides a meaningful improvement in coverage. Anatomical

localization accuracy of 75.62% across detected wounds indicates that the skeleton-freeze projection is reliable when a wound is detected.

Table 4: Full-system scanning evaluation across four sessions (14 wound patches each).

Session	Placed	Detected	Correct Location
S1	14	10	7/ 13
S2	14	10	7/ 10
S3	14	12	9/ 13
S4	14	8	7/ 8
Total	**56**	**40 (71.43%)**	**30 (75.62%)**

6.3 Locking Mechanism Qualitative Results (Simulation)

In Gazebo, the FSM progresses correctly through IDLE, APPROACH, and LOCKING in all tested runs. During steady state with no patient movement, the end-effector holds the 15 cm standoff without triggering corrections, confirming that sensor-induced drift stays below the thresholds. When the wound patch is manually displaced by 2–4 cm to simulate breathing or repositioning, the locking node detects the threshold violation within one median-filter window and issues a Pilz PTP correction that re-establishes alignment within one to two seconds, with no visible overshoot.

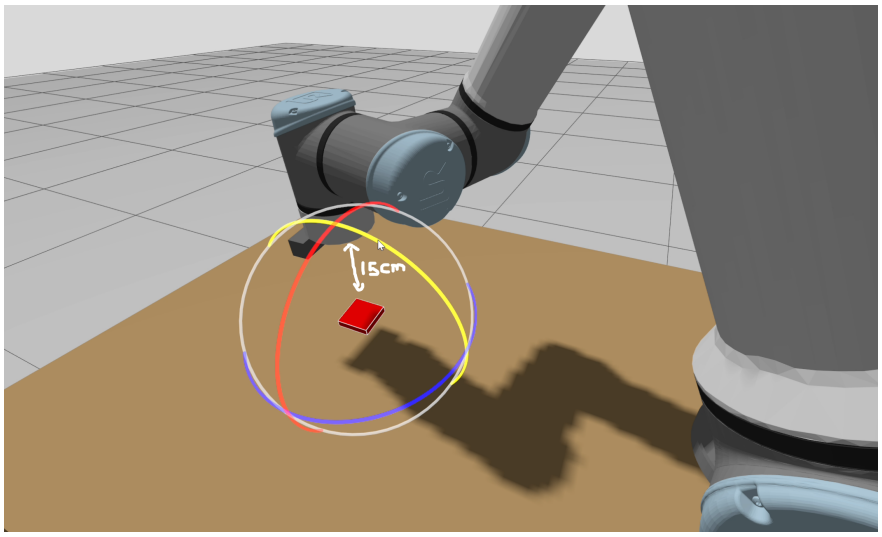

Figure 8: Locking mechanism in Gazebo simulation. The UR16e holds a perpendicular 15 cm standoff to the wound surface. Manual wound displacement triggers a Pilz PTP correction restoring alignment within 1–2 seconds.

6.4 Discussion

High precision, limited recall. Precision of 0.88 is strong for a 140-image training corpus. Recall of 0.63 is the system's most significant limitation, and it is directly tied to dataset size, wound appearance varies widely with lighting, skin tone, and wound morphology, and 140 images cannot adequately cover this space. More training data, and potentially wound-type stratification (laceration, burn, pressure ulcer), would be the highest-impact improvement.

Detection rate vs. localization accuracy. The gap between 71.4% detection and 75.6% localization of detected wounds shows that the skeleton projection is reliable once a wound is found, errors come primarily from YOLO misses rather than from incorrect anatomical assignment. The 0.08 m deduplication radius consistently prevents duplicate entries without merging physically distinct wounds.

SVD plane fitting reliability. The 2σ depth filter before SVD is critical in practice. Without it, boundary pixels where depth wraps between foreground and background corrupt the normal estimate. With it, normals are stable across consecutive frames and produce smooth target frame updates.

Locking stability and threshold design. The 1.5 cm / 8.6° thresholds are well-matched to the Kinect V2's noise floor in simulation, the robot holds still under typical noise while responding promptly to genuine motion. Pilz PTP provides the determinism that OMPL-based planners cannot: corrections arrive within consistent, bounded time windows, which is a prerequisite for any physical treatment deployment.

Kinect V2 range limitation. The 50 cm minimum depth range means the sensor cannot resolve fine surface geometry at the 15 cm treatment standoff. This is the primary hardware constraint blocking physical deployment of Pipeline 2. A short-range depth sensor (e.g., Intel RealSense D435, minimum range $\approx$ 10 cm) would resolve it directly.

7. Conclusion

We built and evaluated a two-pipeline ROS2 system for autonomous wound scanning and treatment on a UR16e manipulator. Pipeline 1, on the physical robot, achieves 71.43% wound detection and 75.62% anatomical localization accuracy across 56 wound placements, producing per-wound clinical reports with location, severity, and confidence. Pipeline 2, in Gazebo simulation, demonstrates stable real-time perpendicular locking with active drift correction, and is architecturally designed so that deploying it on the physical system requires only a launch-file change.

Three engineering decisions proved most consequential: SVD plane fitting with 2σ depth filtering provides robust surface normals from a single-camera setup without any specialized sensor; skeleton state-freezing from a single standoff view enables reliable anatomical context even when the camera loses full-body visibility; and FSM-orchestrated, threshold-gated Pilz micro-corrections make real-time robot locking stable without flooding the motion planner.

Limitations. YOLO recall is bounded by the 140-image dataset. The Kinect V2 minimum

range prevents physical locking deployment without a sensor upgrade. The treatment mechanism is simulation-only at this stage.

Future work. The most direct next steps are deploying the locking pipeline on the physical UR16e with a short-range depth sensor and a fluid treatment effector, and expanding the YOLO dataset with wound type categorization to address recall. Longer term, a context-aware dispenser – selecting treatment formulation based on colorimetric severity and wound type – would complete the autonomous treatment loop this project targets.

References

[1] C. Wang et al., "Fully automatic wound segmentation with deep convolutional neural networks," *Scientific Reports*, vol. 10, p. 21897, 2020.

[2] B. Aldughayfiq et al., "YOLO-based deep learning model for pressure ulcer detection and classification," *Healthcare*, vol. 11, no. 9, 2023.

[3] N. V. Jacob et al., "Automatic wound detection system for multi-ethnic populations using YOLO," in *Proc. CSCE 2024*, ser. CCIS, vol. 2259, 2025.

[4] J. A. A. Velázquez et al., "Automatic wound detection and measurement system in YOLOv7, for future wound healing application," *Nexo Revista Científica*, 2025.

[5] S. Thakar et al., "Area-coverage planning for spray-based surface disinfection with a mobile manipulator," *Robotics and Autonomous Systems*, vol. 158, 2022.

[6] S. Nieto Bastida and C.-Y. Lin, "Autonomous trajectory planning for spray painting on complex surfaces based on a point cloud model," *Sensors*, vol. 23, no. 24, 2023.

[7] H. F. Vargas et al., "Advances towards autonomous robotic suturing: integration of FEM and wound detection through deep learning," *Biomedical Signal Processing and Control*, vol. 101, 2025.

[8] A. Lee et al., "Levels of autonomy in FDA-cleared surgical robots: a systematic review," *npj Digital Medicine*, vol. 7, p. 103, 2024.

[9] Medetec, "Medetec Wound Database," 2013. [Online]. Available: https://www.medetec.co.uk/files/medetec-image-databases.html

[10] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[11] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[12] C. Lugaresi et al., "MediaPipe: A framework for perceiving and processing reality," in *Workshop on Perception for Mobile Robots, CVPR*, 2019.

[13] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," *IEEE Trans. Robotics and Automation*, vol. 5, no. 3, pp. 345–358, 1989.

[14] S. Macenski et al., "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, 2022.

[15] D. Coleman et al., "Reducing the barrier to entry of complex robotic software: a MoveIt! case study," *arXiv:1404.3785*, 2014.