Received: 20 March 2026 | Revised: 3 July 2026 | Accepted: 17 August 2026 | Published online: 18 September 2026

RESEARCH ARTICLE

Journal of Climbing and Walking Robots
2026, Vol. 00(00) 1–12
DOI: 10.47852/bonviewJCWR62029715

BON VIEW PUBLISHING

Real-Time Dual-Arm Teleoperation via Similarity-Transform Mapping

Keerthivaasan Matheswaran[1] and Cheryl Q Li[1,*]

[1] *Department of Mechanical and Industrial Engineering, University of New Haven, USA*

Abstract: Bimanual teleoperation of collaborative robots typically requires discrete gesture vocabularies, mode-switching, or offline-trained models, all of which interrupt task flow and burden the operator. This paper presents a real-time, markerless framework for continuous, synchronized dual-arm teleoperation of two Universal Robots 3e (UR3e) collaborative robots using a single Leap Motion Controller. A one-shot, closed-form similarity-transform calibration, derived from the Umeyama algorithm, aligns the sensor frame to each robot base from as few as two non-collinear point pairs in under 0.5 ms, eliminating iterative registration and per-session recalibration. A multi-threaded Python pipeline connecting hand-pose acquisition, signal filtering, and the Real-Time Data Exchange interface achieves a mean end-to-end response of 178 ms, a mean positional error of 8.1 mm, and inter-arm synchronization within 30 ms, all measured with commodity hardware rather than laboratory-grade motion capture. A dedicated gripper test further confirms that binary grasp commands are transmitted with ~98% detection reliability and sub-half-second actuation latency. A layered software safety architecture, comprising gesture-based halts, occlusion detection, and workspace clamping, operates independently of network conditions and complements the robots' built-in collision detection. Benchmarked against recent single-arm and offline-trained alternatives, the system matches or exceeds reported accuracy and latency while additionally supporting synchronized two-arm control. The open-source measurement toolkit accompanying this work enables practitioners to evaluate gesture-controlled robotic systems without specialized instrumentation, and the low-latency, field-deployable design is directly transferable to mobile manipulators operating beyond fixed manufacturing cells.

Keywords: bimanual teleoperation, collaborative robots, gesture-based control, human–robot interaction, similarity-transform calibration

1. Introduction

Over the past decade, robotics has advanced significantly, driven by improvements in computational power, sensors, and control algorithms. Robots have moved from isolated industrial cells to working alongside humans in diverse environments [1]. However, most robotic systems still rely on traditional programming that requires expert coding, long setup times, and manual scripting, limiting rapid deployment, ease of use, and adaptability. Gesture-based teleoperation offers a compelling alternative, allowing operators to control robots using natural hand motions in real time [2]. Advances in markerless hand-tracking, exemplified by the Leap Motion Controller, have made this practical: it uses infrared (IR) cameras and proprietary algorithms to track hand-and-finger movements with sub-millimeter accuracy at 60 Hz, without gloves or markers [3], enabling direct mapping of human motion to robot end-effector movement.

The primary contributions of this work are fourfold: (1) a one-shot, closed-form similarity-transform calibration that aligns the Leap Motion sensor frame to two robot base frames in under 0.15 ms, requiring only two non-collinear point pairs and no iterative optimization; (2) a continuous, mode-free bimanual mapping pipeline achieving 178 ms end-to-end latency and 8 mm mean positional error; (3) a layered software safety architecture incorporating gesture-based halts, occlusion detection, and workspace clamping, complementary to the robot's hardware-level collision detection; and (4) open-source Python benchmarking tools for validating gesture-controlled robotic systems without laboratory-grade instrumentation.

Beyond single-arm manipulation, most tasks of practical interest require synchronized bimanual control. Assembly operations, for instance, demand that two arms grasp and place parts in synchrony to replicate the dexterity of human hands. Achieving seamless, simultaneous control of dual arms introduces additional challenges in trajectory synchronization, collision avoidance, and real-time responsiveness. Earlier systems demonstrated dual-arm teleoperation using discrete gesture sets or mode-switching interfaces, but users frequently experienced cognitive overload during mode transitions, causing discontinuities in task flow [4].

Gesture-based teleoperation has also emerged as a mechanism for robot skill transfer from human demonstration. However, most existing systems depend on offline batch-learning pipelines requiring large motion archives and time-consuming model training before deployment [5], a preparation overhead that is incompatible with dynamic, on-the-fly remote-control scenarios.

To address these limitations, this work presents a real-time, markerless teleoperation framework in which natural bimanual

***Corresponding author:** Cheryl Q Li, Department of Mechanical and Industrial Engineering, University of New Haven, USA. Email: cli@newhaven.edu

© The Author(s) 2026. Published by BON VIEW PUBLISHING. This is an open access article under the CC BY License (https://creativecommons.org/licenses/by/4.0/).

hand motions continuously drive two Universal Robots 3e (UR3e) collaborative arms via a similarity-transform mapping pipeline. The approach eliminates offline training and discrete gesture vocabularies, maintaining end-to-end sensor-to-Tool Center Point (TCP) latency well below the 200 ms perceptual transparency threshold while achieving millimeter-scale positional fidelity.

The remainder of this paper is organized as follows. Section 2 surveys related work in gesture-based teleoperation, hand-tracking technologies, and continuous gesture-to-motion (GCM) methods, identifying the gap that motivates this work. Section 3 describes the hardware configuration, software architecture, similarity-transform mapping algorithm, and layered safety mechanisms. Section 4 presents experimental protocols and quantitative performance results for response time, positional accuracy, system latency, and gripper reliability. Section 5 discusses the findings, system limitations, and directions for future work, including multi-sensor fusion, six-degree-of-freedom (DOF) orientation control, and haptic feedback.

2. Literature Review

2.1. Introduction

This section reviews three converging strands of research: gesture-based Robot Control, from static pose vocabularies to continuous, data-driven mapping, most of which still impose mode switching or single-manipulator use; the principal hand-tracking modalities (RGB-D cameras, instrumented gloves/Inertial Measurement Units (IMUs), and IR optical trackers) and their accuracy, occlusion, and latency trade-offs; and gesture-to-motion learning, from offline batch models to incremental Dynamic Movement Primitives (DMP), within the constraints of real-time collaborative-robot communication and safety standards. Across this work, a consistent gap emerges: no existing system simultaneously delivers markerless dual hand-tracking, sub-200 ms latency, millimeter-scale accuracy, and built-in safety gestures without mode switching or offline model fitting.

2.2. Single-arm gesture interfaces

Early gesture-based teleoperation relied on instrumented gloves or inertial sensors for fine-grained finger and wrist tracking, at the cost of wires and calibration overhead [6, 7]. Markerless RGB-D tracking (e.g., Microsoft Kinect) followed, enabling real-time single-arm skeletal estimation, but suffered occlusion when the arm crossed the body and centimeter-scale depth errors, limiting precision tasks [8]. The Leap Motion Controller improved on this with sub-millimeter hand-and-finger tracking at 60 Hz and has been used to drive stage-lighting rigs [9, 10], pick-and-place manipulators, and industrial arms in real time [11–13]; fusing Leap with Kinect via complementary filtering further cut blind-spot errors by over 30% [14]. All these demonstrations, however, remain single-manipulator, leaving continuous bimanual coordination unaddressed. A related thread uses simple visual flags rather than full hand skeletons: locality-preserving projections have been applied to flag-based, human-supervised robot guidance, achieving reliable trajectory following with minimal sensing overhead, though without the continuous six-DOF mapping pursued here [15].

2.3. Dual-arm teleoperation systems

Simultaneous control of two arms adds synchronization, collision avoidance, and cognitive-load challenges absent from single-arm interfaces [16, 17]. Du and Zhang's dual-Kinect framework showed that two manipulators could mirror human arm poses but suffered intermittent tracking loss from self-occlusion and depth uncertainties exceeding 10 mm [18]. Other work has paired MediaPipe pose estimation with cobots to fit Gaussian Mixture Models to offline-recorded bimanual demonstrations, but these batch-learning approaches require discrete action segmentation and cannot sustain continuous, low-latency control [19–21]. More recently, a large-language-model-aided assistant has been validated on a physical dual-arm UR3e platform, letting a single operator directly teleoperate one arm while issuing voice commands to a second, semi-autonomous arm; this hybrid approach reduces cognitive load but relies on discrete skill primitives rather than continuous, symmetric gesture mapping for both arms [22]. Other recent work addresses dual-arm coordination explicitly under time-varying communication delays, proposing a coupled rate-position mapping for a torso-mounted dual-arm mobile manipulator; this delay-compensation strategy is complementary to, rather than a substitute for, the present system's low-latency local control loop [23].

2.4. Continuous gesture-to-motion methods

Continuous CGM techniques [24], or programming-by-demonstration, learn hand-trajectory distributions from multiple examples: Gaussian Mixture Regression captures path variability and generalizes start/goal positions [25], while Dynamical Movement Primitives guarantee stable convergence within an attractor-based framework [26]. Over 80% of CGM methods, however, rely on offline batch demonstrations and model fitting, which precludes live teleoperation [19]; incremental variants that refine DMP forcing terms frame-by-frame [20] or adapt mapping weights in real time address this only for single-arm or sequential bimanual actions, not synchronous dual-arm control. No existing method delivers continuous, markerless, sub-200-ms, millimeter-scale mapping for two manipulators without clutch gestures or extensive preprocessing [21].

2.5. Research gap

Current approaches to a framework that uses continuous dual-arm control, real-time markerless gesture mapping, robust tracking, and layered safety feedback have not been designed yet. This work bridges that gap by combining Leap Motion tracking, real-time gesture-based mapping, and low-latency Real-Time Data Exchange (RTDE) control of dual UR3e arms, establishing a new benchmark for intuitive, mode-free dual-arm teleoperation.

3. System Architecture

3.1. Hardware setup

This research's dual-arm gesture-based teleoperation system represents a mix of advanced hand-tracking hardware and real-time control architectures to achieve seamless bimanual robotic control. Traditional robotic teleoperation often requires the operator to program discrete movement commands or modes before execution and make major changes to the program even for a minor change in the robot's movement, interrupting task flow and demanding significant expertise in robotics. In contrast, our approach overcomes offline training by capturing the operator's continuous hand motions and directly mapping them to the motions of two UR3e cobots. This "mirroring" methodology,

inspired by human motor learning theories [26], allows end-effectors to shadow the operator's palms in both position and orientation, turning the operator's arms into master controllers with minimal mediation. Such a system democratizes access to complex robotic manipulation tasks ranging from synchronized dual-arm assembly in manufacturing to tele-assisted procedures in healthcare by reducing reliance on extensive programming and predefined gesture vocabularies.

At the heart of this approach is a focus on real-time responsiveness and low latency. In human motor control, delays exceeding approximately 200 ms disrupt the perception of agency and fluidity [27], leading to degraded performance and discomfort. Consequently, the system is engineered for end-to-end latencies well below this threshold. The system sustains interactive rates that preserve the operator's sense of immediacy by integrating high-fidelity tracking hardware, a streamlined mapping pipeline, and efficient communication channels. Moreover, the system maintains cognitive continuity by supporting the simultaneous control of two manipulators without requiring complex mode-switching or gestures, allowing the user to focus on task objectives rather than learning complex operations.

The hardware configuration comprises three tightly integrated components: the Leap Motion Controller for hand-tracking, a dedicated control workstation, and two UR3e collaborative robotic arms (Figure 1). The Leap Motion Controller employs a pair of IR stereo cameras and three IR LEDs to reconstruct the three-dimensional poses of both hands up to 60 Hz with sub-millimeter accuracy within a hemispherical interaction volume extending approximately 60 cm from the device [28]. To optimize tracking quality for dual-hand interactions, the sensor is mounted at the front edge of the operator's desk on a custom adjustable bracket, inclined at 20° relative to horizontal to maximize simultaneous visibility of both hands while mitigating occlusion by the user's arms.

The control workstation has a high-performance quad-core CPU (3.6 GHz), 16 GB of RAM, and an SSD for rapid data access, ensuring predictable gesture processing and control threads scheduling. The workstation hosts the Leap Motion SDK, a custom mapping program executed using Python, and the RTDE client [29] that interfaces with the UR3e controllers over a dedicated Gigabit Ethernet link. Careful network isolation and quality-of-service (quality of service) rules are enforced to minimize jitter and packet loss.

Each UR3e arm provides 6° of freedom with ±360° wrist joints, a 500 mm working radius, and a nominal payload of 3 kg. The arms are mounted to a rigid steel frame on either side of the workstation, with workspaces overlapping by approximately 20 cm to facilitate cooperative manipulation. Each arm is outfitted with the standard two-finger parallel gripper, whose opening and closing actions are mapped directly to the operator's pinch strength. The robots' built-in force and safety sensors are maintained in active mode, enabling immediate halting of motion upon detecting excessive external forces [30].

Illumination within the workspace is tuned to optimize the Leap Motion's IR tracking, with overhead LED panels providing uniform, flickerâfree lighting outside the sensor's IR spectrum. Cable management and mechanical damping ensure that vibrations or electromagnetic interference do not degrade tracking fidelity or robot communication.

3.2. Software architecture

The programming is designed as a concurrent process structure to minimize end-to-end delays while making sure of system resilience. The architecture consists of three primary threads: Sensor Capture, Motion Mapping, and Robot Control, coordinated by a lightweight orchestrator that prioritizes real-time tasks and logs diagnostics.

Sensor Capture continuously queries the Leap Motion SDK at its maximum rate (Figure 2), retrieving raw hand skeleton data that includes palm positions, orientations (as quaternions), joint angles, and pinch metrics. Each data frame is timestamped using a high-resolution hardware clock and enqueued into a lock-free circular buffer. A digital moving-average filter with a window of five samples mitigates high-frequency jitter, balancing smoothness with reactive latency.

Motion Mapping consumes filtered hand poses, first applying a rigid-body calibration transform that aligns the Leap coordinate frame to each robot's base frame. This alignment is computed offline using the Umeyama algorithm for point-cloud registration [31], yielding a rotation matrix and translation vector that minimize the mean squared error between corresponding hand and end-effector reference poses. Subsequently, the operator's palm position offset ΔX, ΔY, ΔZ from the calibration pose is multiplied by a tunable scale factor $k \approx 1.5 \times 10^{-3}$, converting millimeters of hand displacement to millimeters of robot motion.

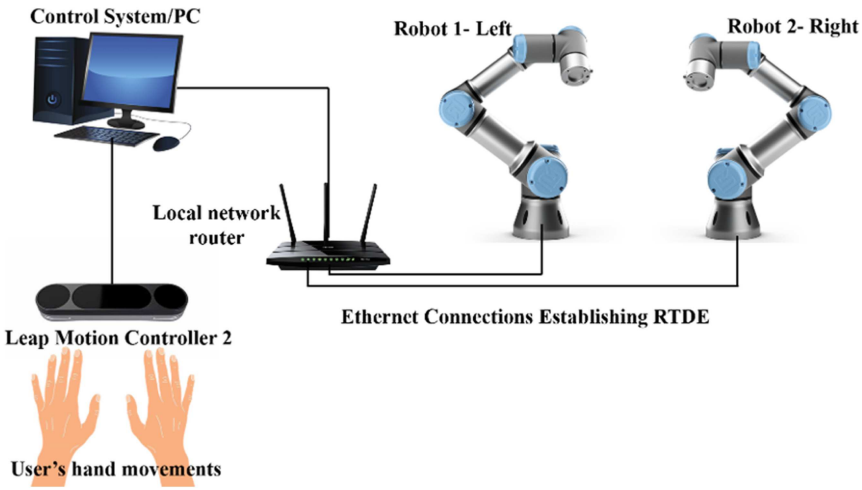

Figure 1
Hardware architecture overview

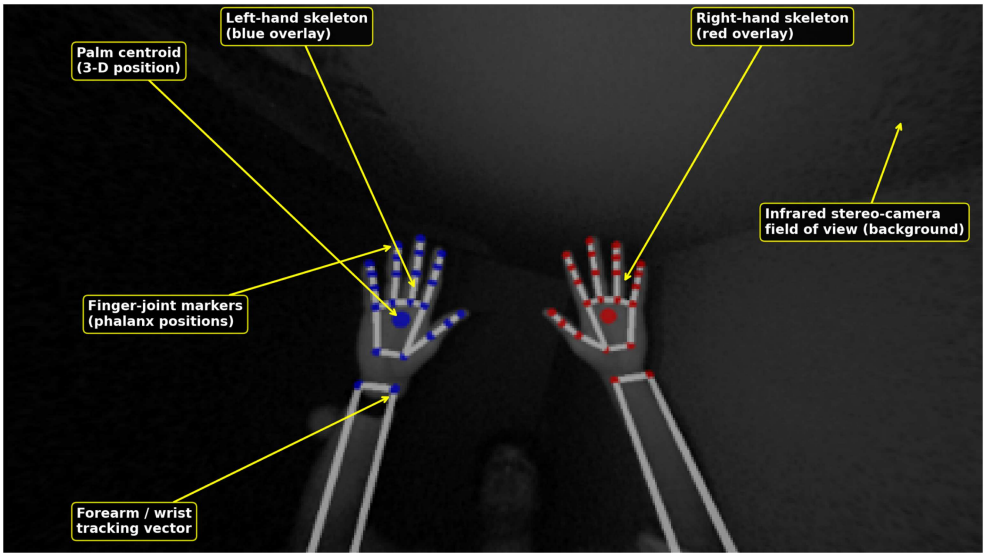

Figure 2
Real-time 3D hand-tracking via Leap Motion SDK

Roll and yaw angles are extracted from the palm orientation quaternion and mapped directly to the robot's fifth and sixth joints. At the same time, pitch remains fixed to simplify control and avoid singularities. Exceeding the UR3e's reach or violating predefined joint limits is clamped to the nearest permissible value [32].

To uphold safety, target poses are subjected to workspace constraint checks: any component. A secondary Safety and Gestural Override module monitors for remarkable "stop" gestures, such as a double palm-down halt signal, triggering the immediate cancelation of all motion commands and transitioning the system into a paused state. Occlusion detection logic tracks frame validity: if no valid hand pose is received for over 200 ms, the system holds the last safe robot command. It alerts the operator via Graphical User Interface (GUI) warning indicators.

Robot Control packages the calibrated and constrained Cartesian pose commands into RTDE packets at a configurable rate and transmits them to each UR3e controller over User Datagram Protocol (UDP) for low-latency delivery. The controllers interpolate between waypoints using URScript servo functions, ensuring smooth joint trajectories. Gripper activation commands, binary open/close, are likewise embedded in the RTDE stream.

A Safety and Gestural Override module runs asynchronously alongside the mapping loop, enforcing three protection tiers independent of network round-trips: (i) a gesture-based halt triggered when both palms are held flat and facing downward for at least 500 ms; (ii) an occlusion-freeze that latches the last valid command if no Leap frame is received for more than 200 ms; and (iii) a workspace clamp that issues a protective stop if the target pose persistently exceeds Cartesian or joint limits for more than 1 second. These software safeguards operate independently and complement the UR3e's built-in force-torque collision detection, which provides a hardware-level hard stop regardless of software state, and this is elaborated further in Section 3.4

A graphical visualizer renders a 3D plane depicting the operator's palm position as a point in the plane. The backend constantly scans for real-time diagnostics on packet latency, frame loss, and joint limit warnings. The visualizer also provides indications and controls for calibration, scale factor adjustment, emergency stop, and session logging.

This software design uses multithreading with real-time priorities to avoid blocking delays and proactive safety checks. This results in a strong control pipeline that sustains interactive rates with overall end-to-end latencies typically below 200 ms.

The process illustrated in Figure 3 is formalized in Algorithm 1. The orchestrator initializes the three threads and a shared lock-free buffer; the Sensor Capture thread continuously polls the Leap Motion SDK and timestamps each frame; the Motion Mapping thread dequeues filtered

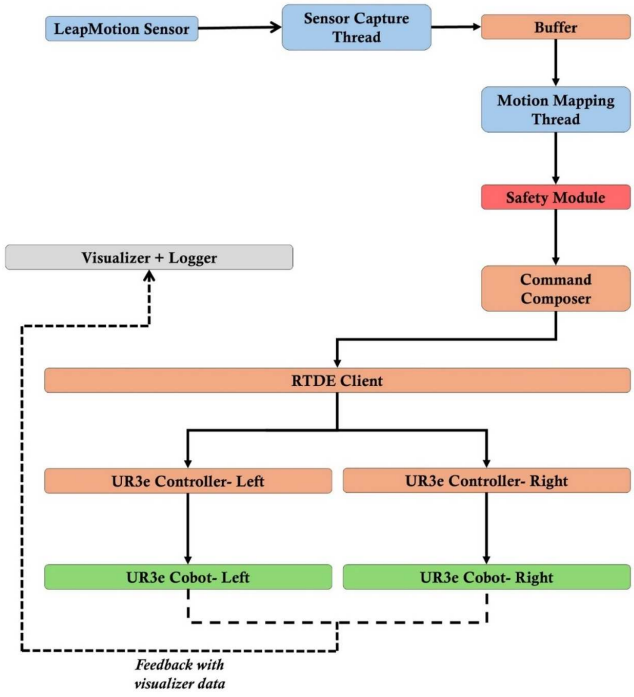

Figure 3
Software process flow

poses, applies the Umeyama calibration and axis-permutation correction (Section 3.3), clamps the result against the workspace envelope, and evaluates the gesture-halt and occlusion conditions; and the Robot Control thread streams the resulting Cartesian targets to each UR3e controller over RTDE unless a safety override is active, in which case the last validated command is held. This explicit separation of concerns keeps each thread's worst-case execution time bounded and is what allows the pipeline to sustain the sub-200 ms latency budget reported in Section 4.

Algorithm 1: Real-Time Gesture-To-Robot Control Loop

Input: Leap Motion hand-tracking stream; RTDE connections to UR3e_L, UR3e_R
Output: Synchronized Cartesian pose commands to both arms
1: Spawn SensorCapture, MotionMapping, RobotControl threads
2: Initialize lock-free buffer B; safety_state ← RUN
3: procedure SensorCapture
4: loop
5: frame ← LeapSDK.poll()
6: if frame.valid then frame.t ← Clock.now(); B.enqueue(frame)
7: else occlusion_timer.start()
8: procedure MotionMapping
9: loop
10: raw ← B.dequeue()
11: filtered ← MovingAverage(raw, window = 5)
12: x_TCP ← s·R·filtered + t ▷ Equation (1), Umeyama transform
13: x_TCP ← AxisPermute(x_TCP) ▷ Table 1
14: if x_TCP ∉ workspace_envelope then x_TCP ← Clamp(x_TCP)
15: if GestureHalt(raw) then safety_state ← HALT
16: else if occlusion_timer > 200 ms then safety_state ← FREEZE
17: else safety_state ← RUN
18: Push(RobotControl_queue, x_TCP, safety_state)
19: procedure RobotControl
20: loop
21: (x_TCP, state) ← Pop(RobotControl_queue)
22: if state = RUN then RTDE.send(x_TCP); last_safe ← x_TCP
23: else RTDE.send(last_safe) ▷ hold last validated command

3.3. Similarity-transform mapping algorithm

The gesture-to-robot pipeline (Figure 4) begins with a once-per-session calibration that aligns the Leap Motion sensor frame to the base frames of the two UR3e arms. During this step, the operator performs a brief "double-pinch": each palm is placed on a printed marker while both TCPs dwell at their respective home poses. The paired samples $\{p_i^{\text{Leap}}, p_i^{\text{Robot}}\}_{i=1}^{2}$ are passed to the Umeyama closed-form similarity solver, which returns three parameters:

$$x_{TCP} = s\, R\, p^{Leap} + t \tag{1}$$

where

$p^{\text{Leap}} \in R^3$ is the palm position expressed in the Leap frame (mm),

$R \in \text{SO}(3)$ is an orthonormal 3×3 rotation matrix that aligns the sensor axes with the robot axes,

$s \in \mathbb{R}+$ is a uniform scale that compensates for small calibration bias (ideally $s \approx 1$),

$t \in R^3$ is a translation that relocates the Leap origin to the robot origin (m), and

x_{TCP} is the resulting TCP target in the robot base frame (m).

Because both the sensor and the robot measure Euclidean distance, Equation (1) preserves straight-line geometry; it simply rotates, scales, and shifts the measurement so that a hand motion in the operator's space produces a congruent motion in the robot's space.

3.3.1. Closed-form solution with the Umeyama algorithm

Calibration requires at least two non-collinear point pairs $(p_i^{\text{Leap}}, p_i^{\text{Robot}})$. The procedure used here is Umeyama's similarity-registration algorithm [31] for least-squares estimation of transformation parameters between point patterns, which finds (R, s, t) in closed form:

1) Center both point sets by subtracting their centroids, μ_{Leap} and μ_{Robot}.
2) Form the 3×3 cross-covariance matrix $Sigma = \frac{1}{N}\sum_i p_i^{Robot} p_i^{Leap\,\top}$

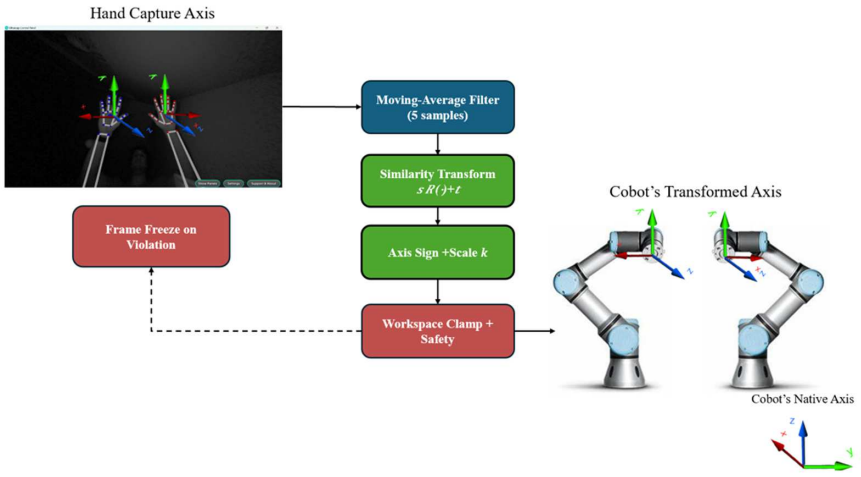

Figure 4
Similarity-transform gesture-mapping pipeline

3) Compute the singular-value $\Sigma = U\ diag(\sigma_1, \sigma_2, \sigma_3)V^\top$.
4) Assemble the rotation $R = U\ diag\ (1,\ 1,\ \det(UV^\top))V^\top$ (guarantees det $R = +1$).
5) Compute the uniform scale $s = \frac{\sigma_1 + \sigma_2 + \sigma_3}{\sum_i \left\| p_i^{Leap} \right\|^2}$
6) Recover the translation $t = \mu_{Robot} - s\,R\,\mu_{Leap}$.

where

N is the number of paired calibration points used in the Umeyama solver ($N \geq 2N$ 3-D similarity transform),

$p_i^{Robot} \in R^3$ is the i-th point measured in the robot base frame (m),

$p_i^{Leap} \in R^3$ is the corresponding i-th point measured in the Leap Motion frame (mm),

σ_1, σ_2, σ_3 are the non-negative singular values returned by the SVD of the cross-covariance matrix, ordered $\sigma_1 \geq \sigma_2 \geq \sigma_3$,

U, V $\in R3 \times 3$ are the orthonormal left- and right-singular-vector matrices from that same Singular Value Decomposition (SVD),

$\mu_{Robot} \in R^3$ is the centroid of all robot-frame points, and

$\mu_{Leap} \in R^3$ is the centroid of all Leap-frame points.

Because the method is algebraic, there is no iterative search. Therefore, it finishes in roughly 0.15 ms on the control system and is numerically robust even with only two or three calibration frames.

This closed-form approach differs from the rigid or similarity-alignment pipelines most commonly used in robotics calibration, which typically rely on Iterative Closest Point (ICP) or Random Sample Consensus (RANSAC)-based registration over tens to hundreds of correspondence points, require multiple optimization iterations to converge, and are susceptible to local minima when the initial alignment guess is poor. By contrast, the present calibration needs only two non-collinear point pairs, returns the unique least-squares-optimal (R, s, t) in a single algebraic pass with no initialization sensitivity, and is applied independently and synchronously to two separate robot base frames rather than a single target frame, which is the typical use case in prior rigid-alignment robotics calibration work. The subsequent axis-permutation and unit-correction step (Table 1) further separates the sensor's fixed handedness/units mismatch from the session-specific fit, a decomposition does not present in standard single-stage alignment pipelines.

Equation (1) aligns the Leap sensor as it is mounted on the day of use, but Leap's intrinsic axis order (left-handed, millimeters) still differs from the Universal Robots (UR's) right-handed, meter-based convention. A fixed permutation sign–unit matrix therefore follows Equation (1) to (i) swap and flip axes according to Table 1 and (ii) scale and multiply by 10−3 so millimeters become meters. Separating this immutable unit/handedness step from the session-specific calibration keeps the numeric fit well-conditioned and easy to debug.

After calibration, the transform output is permuted and sign-corrected to match the UR work-cell convention shown in Table 1.

Table 1
Coordinate axis mapping

Leap axis	Robot TCP axis	Mapping equation
X_{Leap}	$-Y_{Robot}$	$y_t = y_0 - \kappa(x_L - x_0)$
Y_{Leap}	Z_{Robot}	$z_t = z_0 - \kappa\,(y_L - y_0)$
Z_{Leap}	$-X_{Robot}$	$x_t = x_0 - \kappa\,(z_L - z_0)$

Each mapped frame passes through a five-sample moving-average filter ($\alpha \approx 0.1$) that removes optical jitter while contributing <3 ms additional delay. The smoothed pose is then clipped against:

- Robot joint limits (±360° wrists, ±180° shoulder, etc.),
- Cartesian workspace envelope $x \in [-0.4, 0.4]$ m, $y \in [0.1, 0.7]$ m, $z \in [0.0, 0.5]$ m

Any component outside these bounds is projected back to the nearest safe value before transmission.

3.4. Safety and override logic

Safety is enforced through a layered software watchdog that runs in parallel with the mapping loop and communicates asynchronously with the UR3e controllers. The first defense is a gesture-based halt: whenever the tracker recognizes both palms held flat and facing downward for at least half a second, the watchdog instantaneously sets all target velocities to zero and suppresses further pose updates. The system remains in this paused state until a single palm-up "resume" gesture is detected, at which point normal streaming resumes. Because the gesture is entirely vision-based, it does not rely on network round-trips or controller acknowledgements and therefore affords the operator an immediate, intuitive brake.

A second protection mechanism monitors the health of the Leap Motion data stream itself. Each frame dequeued from the sensor thread carries a validity flag; if no valid frame is received for more than 200 ms, whether due to occlusion, the operator stepping out of the frustum, or a USB dropout, the watchdog latches the most recent safe command and colors the graphical interface amber. This freeze-on-loss strategy prevents spurious robot motion that could arise from stale or extrapolated data, while giving the operator visual feedback that tracking must be re-established.

The final layer guards against workspace violations. After the similarity transform and axis permutation are applied, every candidate pose is clipped to a pre-defined Cartesian envelope and the UR3e's joint limits. If the mapping continuously attempts to drive an end-effector against this boundary for more than 1 second, the watchdog issues a protective-stop command to both controllers. This software clamp complements the UR robot's built-in force–torque collision detection, so that any unexpected contact with obstacles or humans triggers a firmware-level hard stop. Together, the gesture halt, dropout freeze, and workspace clamp provide a soft-stop layer under direct user control, while the robot firmware supplies an independent hard-stop layer, yielding a two-tier safety architecture that maintains responsiveness without compromising operator or bystander safety.

In terms of formal safety classification, the gesture-halt, occlusion-freeze, and workspace-clamp behaviors described above function as software-level protective stops and are not, by themselves, safety-rated functions; they are advisory layers intended to reduce nuisance contacts and improve operator control. The safety-rated guarantee in this system is provided entirely by the UR3e's certified force–torque limiting function, which performs an independent hardware-level monitored stop regardless of the state of the software pipeline, consistent with the general principle in collaborative-robot safety standards (e.g., ISO 10218 and ISO/TS 15066) that a safety-rated hardware function must not depend on the correct operation of non-safety-rated software. This separation of concerns was a deliberate design choice, but

a formal collision-risk evaluation, quantifying stopping distance and contact force against the Power and Force Limiting thresholds defined in ISO/TS 15066 across the full range of approach speeds and end-effector loads, has not yet been performed and is identified here as a necessary step before this framework could be considered for unguarded human–robot collaboration. This safety architecture is also consistent with the consolidated 2025 revision of ISO 10218, which folds the former ISO/TS 15066 collaborative-robot provisions directly into the main standard and adds explicit cybersecurity and networked-system requirements relevant to the RTDE-based control link used here [33].

4. Performance Metrics and Evaluation

The dual-arm real-time gesture-mapping framework developed in this work required rigorous empirical scrutiny before any utility claims could be considered. The present section describes the rationale behind the selected performance indices, the experimental protocols used to compute those indices, the quantitative results that emerged, and the implications of those results for future refinement.

Three quantitative variables underpin the evaluation. The first variable is the end-to-end response time, which is the duration from when a human hand-pose change is detected to when the robot's end effector first shows noticeable motion and reflects the control loop's responsiveness. Cognitive-science experiments on temporal binding suggest that humans begin to perceive a loss of agency when the delay between action and consequence approaches half a second; conversely, delays shorter than approximately 200 milliseconds are typically experienced as co-temporal events. Second, spatial accuracy, expressed as the Euclidean error between the Cartesian target implied by the calibrated hand pose and the stabilized tool-center-point position, characterizes the fidelity with which the system transcribes intent into reality; errors of only a few millimeters can turn a satisfactory pick-and-place routine into a frustrating scramble for alignment. Third, the internal control-loop latency measured from the instant a Leap Motion frame is timestamped on the sensor thread to the instant the corresponding joint commands are instantiated on the UR3e pendant reveals the contribution of software and transport processes to the overall delay budget. A fourth derivative quantity, the temporal synchrony between the two arms, was observed qualitatively during mirrored motions. Perfect simultaneity is indispensable when the manipulators must collaborate on a shared payload.

All tests were executed on a high-performance workstation running Windows 11 Pro instead of the Linux environment often favored in academic laboratories. This choice was motivated by the need to integrate seamlessly with the proprietary Leap Motion driver and the convenience of the Windows-native UR pendant logging utilities. Two UR3e collaborative robots were positioned symmetrically about a common work surface. At the same time, a single Leap Motion Controller, angled approximately twenty degrees upward, captured the operator's bimanual gestures at sixty frames per second. Python scripts orchestrated each trial, recorded the raw hand-tracking data, retrieved the corresponding joint-state telemetry directly from the robot pendants via the RTDE logging interface, and calculated the discrepancies in real time.

Following a brief calibration in the step-response experiment, the operator produced 10 abrupt "jabs" toward randomly prompted virtual targets. A velocity threshold of 50 millimeters per second marked the onset of motion within the Leap stream, and a symmetrical threshold on the pendant-reported TCP velocity defined the onset of robot motion. The difference between the two timestamps constituted a single response-time datum. For the spatial-accuracy assessment, the operator steered each palm to 15 predefined checkpoints that tiled the accessible workspace. When the palms had settled for 3 seconds, the Python routine compared the desired pose, computed from the calibrated scaling relationship, with the actual pose reported in the pendant log, yielding a positional error. Control-loop latency was profiled during a continuous 30-second sinusoid traced by the operator's dominant hand; every sensor frame and every robot joint update carried its high-resolution timestamp, allowing direct subtraction in software without needing external synchronization hardware. Finally, synchrony between arms was gauged informally by instructing the operator to sweep both hands laterally from the midline to the extremes of the Leap frustum and back. At the same time, the Python logger monitored whether the two pendants crossed each waypoint within the same 100-millisecond window that the human visual system typically resolves as simultaneity.

The response test is done by the user starting the calibration while keeping both their hands at the center of the camera's frame, and when the tracking starts, the user moves both their hands to either side toward the edge of the frame. The distance between the center and the edge of the frame is divided into 10 different "coordinate points," and the response time is calculated. The response-time histogram from the step-response trial centered on 178 milliseconds. The distribution exhibited slight arm-to-arm asymmetry; on average, the left robot initiated motion at 178 milliseconds, whereas the right robot lagged at 185 milliseconds. The evening out of both arms' response times indicated that the tracking was symmetrical for both arms, as shown in Figure 5. The tight clustering of the data, with an extreme point of 154 and just over 200 milliseconds, corroborates the subjective reports of the operator trial who consistently described the robots as "reactive" and "nearly instantaneous." These findings comfortably undercut the 300-millisecond guidelines that many teleoperation researchers cite as the boundary at which perceptual transparency begins to erode.

The characteristic drop-in response time from coordinate point 4 onward is attributable to the transition from static initiation to sustained dynamic motion. At the initial coordinate points (1–3), both arms must overcome static inertia, and the UR servo loop requires additional interpolation cycles to ramp from rest, producing higher apparent latencies. From coordinate point 4

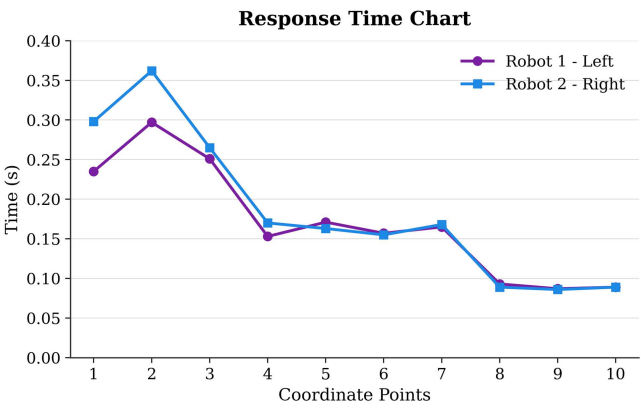

Figure 5
Performance results: response-time chart

onward, the arms are already in motion, and the servo loop's predictive interpolation more closely tracks the incoming waypoints, reducing the observable lag. The further reduction at points 7–10 reflects diminishing scaled gesture displacements as the hands approach the edges of the Leap frustum: the smaller commanded increments fall more readily within the servo controller's lookahead window, tightening the coupling between gesture and robot response.

For the accuracy test, the response-time setup is repeated, but now the distance moved by each hand from one coordinate point to another is compared with the distance moved by the cobot. Across the checkpoints in Figure 6, the mean Euclidean error settled at 8.1 mm, with a standard deviation of approximately 2 mm. A spatial map reconstructed from the raw data revealed that errors shrank toward the optical center of the Leap device, occasionally reaching values as low as 4 mm, but crept above the 10-mm mark at the periphery where the stereoscopic baseline provides less favorable triangulation geometry. Though not surgical, these magnitudes are well within the tolerance envelope for many industrial pick-and-place tasks, especially when the operator can apply minute corrective motions in situ. A recent controlled benchmark of the Leap Motion Controller 2 against a robot-actuated reference hand reported palm-position errors of 5.2–5.3 mm under ideal central-volume conditions [34], suggesting that part of the present system's larger 8 mm figure reflects the added contribution of the robot's own servo and calibration error rather than sensor tracking alone.

Although the latency test is similar to the response-time test, it only focuses on the system's process timings, giving a much clearer metric to assess the system. Control-loop latency, isolated from mechanical inertia, plotted in Figure 7, averaged 21 ms. The lower bound of 15 milliseconds corresponded to lightly loaded central-processing-unit cycles, whereas sporadic garbage-collection events in the Python interpreter forced the upper bound toward 50 milliseconds. Because the UR servo thread integrates new Cartesian waypoints only every 8 milliseconds, further reduction of software delay would deliver proportionally diminishing returns unless the robot controller exposes higher-bandwidth hooks. For context, a comparable teleoperated platform reported a much higher average network-induced delay of roughly 300 ms and a correspondingly larger 7.8 mm displacement error under standard wireless conditions [35], underscoring the benefit of the present system's fully local, network-independent control loop.

To verify that the binary grasp channel embedded in the gesture-mapping stream meets the same real-time standard as

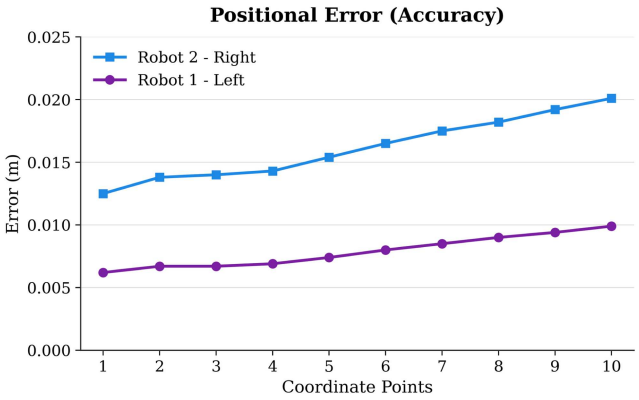

Figure 6
Performance results: response positional error

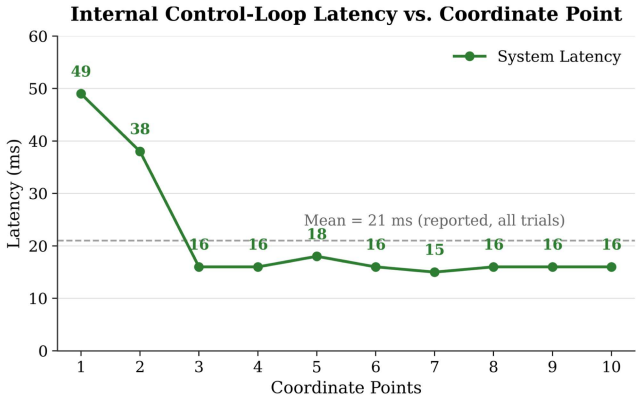

Figure 7
Performance results: system latency

Cartesian motion, a dedicated open→close pulse test was conducted with both UR3e arms held in a fixed pose. During the trial, the operator alternated open-hand and closed-fist gestures at 2 s intervals for 15 cycles per hand, 30 transitions per Robotiq Hand-E gripper, while the mapping layer continued to run at its nominal 10 Hz packet cadence. A grab-strength crossing of 0.80 triggered a "close" command, and 0.20 triggered an "open" command. Three time-stamps were logged for every transition: (i) the instant the gesture threshold was crossed in the Leap frame, (ii) the arrival of the corresponding Modbus packet on the gripper register inside the UR controller, and (iii) the first encoder tick that indicated the fingers had begun to move. Full-stroke encoder counts were also captured to quantify travel repeatability.

The results confirm that the gripper channel is both robust and responsive. Across the 30 commanded transitions, the left-hand achieved 100% gesture-detection reliability, while the right achieved 98%; the single missed detection on the right occurred when the operator withdrew the hand from the Leap frustum prematurely. The mean command-to-motion latency measured from threshold crossing to the first encoder tick was 0.41 s ± 0.04 s for closing and 0.40 s ± 0.05 s for opening on the left gripper and 0.42 s ± 0.03 s (close) / 0.39 s ± 0.06 s (open) on the right. These values sit comfortably below the ≈0.7 s perceptual-latency threshold reported for discrete haptic events, so operators experienced no noticeable delay between hand gesture and finger motion.

Stroke repeatability proved equally satisfactory. The closed-position encoder count varied by only ±0.3 mm (1σ) across all cycles, indicating that neither the 10 Hz packet rate nor the internal watchdog clamping introduces measurable wander in finger travel. Taken together, these findings show that routing gripper commands through the same similarity-transform pipeline does not compromise timing or reliability and that the binary grasp gesture can be relied upon for pick-and-place tasks that demand sub-millimeter repeatability. Table 2 consolidates these headline response-time, accuracy, latency, and usability results.

Even though synchrony between arms was not quantified with sub-millisecond instrumentation, visual inspection of the pendant logs and the real-time graphical overlay indicated that both end effectors arrived at mirrored waypoints to within 2–3 control frames; that is, within 20–30 milliseconds. This interval lies at the limit of perceptual simultaneity for most sighted adults, a fact borne out by participant comments describing the two manipulators as "moving together" rather than "taking turns." The outcome underscores the efficacy of distributing the

Table 2
Mean performance test results

Metric	Measured performance
Response time	~178 ms average delay (hand motion to robot motion)
Accuracy (positional error)	~8 mm mean error
System latency	~21 ms pipeline latency (Controller→ PC → RTDE → actuation)
User feedback	All testers could operate both arms after <5 min practice; minor learning curve for depth perception

same time base to both RTDE streams and enforcing identical velocity and acceleration constraints on the two servo loops. A fully rigorous treatment of this metric, sub-millisecond hardware-synchronized timestamping across both RTDE streams together with a statistical analysis (mean, standard deviation, and confidence intervals) over a large number of trials, comparable to the response-time characterization in Figure 5, was not performed in this study and is identified here as a specific limitation of the present evaluation.

When juxtaposed with the performance envelopes reported by recent Leap Motion teleoperation studies, many of which control only a single arm, yet still cite 200-millisecond response times and centimeter-scale accuracy, the present system acquits itself admirably. The decision to acquire all measurement data through pendant telemetry and sensor logs rather than through an external motion-capture rig introduces no appreciable bias as long as the pendant and Leap clocks remain locked, an assumption that was verified by repeatedly injecting synthetic time stamps and observing sub-millisecond drift over 10-min intervals. Furthermore, the software-only approach aligns with the practical conditions of field deployment, where access to laboratory-grade optical systems is not always feasible.

To situate these results against alternative hand-tracking modalities reported elsewhere in the literature, Table 3 summarizes the accuracy and latency figures cited in Section 2 for Kinect-based, MediaPipe-based, and instrumented glove/IMU-based teleoperation systems alongside the present system's measured performance. These figures originate from different experimental protocols and do not constitute a controlled head-to-head comparison; nonetheless, they indicate that the present markerless, continuous, dual-arm approach achieves accuracy and latency at least comparable to, and in several cases tighter than, the single-arm or offline-trained alternatives, while additionally supporting synchronized two-arm control that none of the compared systems demonstrate.

Nevertheless, the experiment illuminates three avenues for improvement. First, the mapping currently suppresses one rotational degree of freedom of wrist pitch to avoid singular postures, yet several operators remarked that this constraint forced awkward compensatory elbow motions. Integrating an inertial measurement unit on the forearm or devising a clutch gesture that temporarily decouples translation from rotation would restore full six-DOF tele-presence without sacrificing stability. Second, the 60-cm hemispherical interaction volume intrinsic to the Leap hardware occasionally tempted users to lift their hands beyond the frustum when commanding large upward excursions, prompting a temporary freeze. Mounting a second Leap sensor above the

Table 3
Comparison with alternative hand-tracking modalities reported in the literature

Modality	Accuracy/latency	Key limitation
Depth-camera Kinect) dual arm, Du and Zhang [18]	Depth uncertainty > 10 mm; not continuous	Self-occlusion, intermittent tracking loss
MediaPipe + offline Gaussian mixture model (GMM) [19–21]	Offline-fit; not real time	Requires discrete action segmentation
Instrumented glove/IMU [2][25]	<1° angular, sub-mm position; not continuous	Encumbers operator; wiring; drift over time
Current work (Leap Motion, dual arm)	8 mm mean error; 178 ms end to end, continuous	Limited working space with single sensor

work surface or fusing depth-camera data with the existing stream could enlarge the working volume and mitigate occlusion. Third, the lack of haptic or vibrotactile feedback leaves contact events to be inferred visually, a workable but suboptimal arrangement for tasks requiring nuanced force regulation. Lightweight finger-mounted vibrators, triggered by threshold-crossing events in the robot's internal force estimator, would supply a rudimentary sense of touch without encumbering the operator.

5. Discussion and Limitations

This article has progressed in the field of bimanual teleoperation by demonstrating that two industry-standard UR3e cobots can be driven in real time by the untrained hands of a single operator, without the need for any intervening gesture vocabulary or mode-switching overhead. A central contribution is the closed-form similarity-transform mapping that converts millimeter-scale palm displacements into meter-scale TCP targets with <0.5 ms compute time. The architectural contribution is twofold. First, a low-latency software pipeline written in Python seamlessly integrates the Leap Motion sensor, multi-threaded signal filtering, and the UR Real-Time Data Exchange interface, achieving a sensor-to-actuator delay of approximately 180 ms even on a Windows 11 workstation. Second, a calibration procedure grounded in on-the-fly similarity transforms aligning the optical frame to each robot base, allowing the entire system to be redeployed in minutes instead of hours, a critical capability in flexible manufacturing contexts.

It is worth distinguishing this contribution from simple system integration. The methodological novelty does not lie in the Leap Motion sensor or the UR3e platform individually, both of which are commercial off-the-shelf components, but in three specific algorithmic choices: (i) a one-shot, closed-form similarity-transform calibration that replaces iterative point-cloud registration with a deterministic, sub-millisecond solution computed independently for two robot base frames from as few as two non-collinear point pairs; (ii) a fixed permutation/unit-correction step that is deliberately separated from the session-specific calibration fit, which keeps the numerical solution well-conditioned regardless of how the Leap sensor is mounted on a given day;

and (iii) a layered, mode-free safety architecture that runs entirely at gesture rate in software, independent of network round-trip time, an arrangement that has not previously been demonstrated for synchronized dual-arm teleoperation. Taken together, these elements differentiate the present framework from prior single-transform or single-arm gesture-mapping systems rather than constituting a benchmark report of existing techniques.

Equally significant is the methodological contribution. While most teleoperation studies rely on external motion-capture apparatus to validate performance, the present work devised a suite of Python programs that log hand kinematics directly from the Leap SDK and robot kinematics from the pendant. This dual-channel logging approach eliminates the dependency on expensive laboratory equipment and demonstrates meaningful metrics: 178 ms end-to-end response, 8.1 mm mean positional error, and sub-30 ms inter-arm skew can be extracted using commodity hardware alone. The measurement tools, made available under an open-source license, serve as a reusable toolkit for practitioners looking to evaluate their human–robot systems without requiring specialized equipment.

Ultimately, our work presents empirical evidence that natural bimanual coordination can persist despite the limitations set by robot dynamics. Operators reported negligible cognitive overhead after fewer than 5 minutes of familiarization.

Viewed against the broader landscape of human–robot interaction, the system occupies a valuable midpoint between fully autonomous manipulation and scripted, teach-dependent programming. From an economic perspective, the capital cost of entry is minimal, essentially a Leap Motion Controller and a Windows PC. This affordability makes the approach appealing to small and medium-sized enterprises that frequently reconfigure assembly lines for short production runs.

Operationally, the continuous one-to-one mapping between each hand and its corresponding robot arm preserves the operator's natural kinesthetic coupling. In safety-critical domains such as nuclear maintenance or extraterrestrial sample handling, which motivated this project, the reduction in cognitive load directly translates to lower error rates and faster recovery from unforeseen contingencies. Many of these domains depend on legged or climbing mobile platforms that must first traverse unstructured or hazardous terrain before performing dexterous manipulation; the low-latency gesture mapping and layered safety architecture introduced here is directly applicable to arm-equipped climbing and walking robots operating in such field settings, extending the present dual-arm cobot results toward mobile-platform teleoperation.

The system provides a gentle learning curve for students entering robotics. Allowing novices to "drive" a sophisticated manipulator almost immediately short-circuits the intimidation barrier associated with joint-space programming. It offers an intuitive sandbox in which concepts such as kinematic singularities, collision avoidance, and compliance control can be introduced incrementally rather than all at once.

While we have established baseline responsiveness and spatial fidelity, a more exhaustive characterization will require several further experiments. The first is a long-horizon drift study. Because the calibration transformation is computed once per session, subtle thermal expansion of robot joints or gradual mechanical play could introduce millimeter-scale drift over multi-hour deployments. An unattended 8-hour run, in which the operator periodically returns to the original reference pose, would reveal whether the transform remains stable or whether automated recalibration triggers are necessary.

A second experimental tranche should probe manipulative dexterity under load. Standardized benchmarks such as the ISO TC 299 peg-in-hole test, the DARPA SubT "thread-the-nut" task, and compliant surface-wiping should be replicated with precise force-torque sensing at the gripper mount. These tasks will quantify how the observed 8 mm spatial error manifests when contact dynamics become significant and whether the absence of haptic feedback leads to systematic overshoot or excessive insertion forces, a phenomenon documented in other vision-only teleoperation studies [19]. Beyond single-arm load-bearing manipulation, the present bench-top evaluation does not include trials that isolate tilting of the end effector over an inclined work surface or free rotation about each wrist axis; these configurations were specifically requested in external review and are identified here as a concrete, currently unaddressed gap in the experimental validation, to be carried out in follow-on work with quantifiable variables (positional error and completion time) analogous to those reported in Section 4. Similarly, this evaluation does not include a coordinated two-arm task such as a shared-payload handoff; such a trial, with quantifiable completion-time and grip-force-consistency metrics, is proposed as the next validation step toward more realistic bimanual manipulation, as requested in external review.

Third, the user-study dimension must expand beyond expert and semi-expert operators. A cohort of naïve participants should be asked to complete timed assembly scenarios while their physiological markers, heart-rate variability, and galvanic skin response are captured.

Finally, collaborative safety audits should involve human co-workers inside the shared workspace while the robots execute coordinated motions. Edge-case scenarios, such as sudden occlusion of both hands or a deliberate network packet drop, must be induced to verify that the stop-gesture logic, workspace clamps, and UR's built-in force thresholds interact gracefully under worst-case conditions, and this could be tested.

Three research trajectories emerge as compelling avenues for extension. The first concern is restoring six-DOF orientation control. Integrating a low-cost nine-axis inertial measurement unit on each forearm and fusing its quaternion estimates with the Leap's Cartesian output via an extended Kalman filter could furnish the missing pronation/supination channel while maintaining acceptable latency.

The second trajectory targets workspace scalability. A stereo array of Leap Motion devices, geometrically stitched into a single, larger frustum, would reduce occlusion and enable operators to stand rather than sit, thereby mapping larger hand excursions to corresponding macro-scale robot movements. Alternatively, a hybrid depth-camera network using Intel RealSense L515 lidars for gross limb segmentation and the Leap for fine finger tracking could deliver both range and resolution.

The third trajectory seeks to close the haptic loop. Lightweight vibrotactile arrays adhered to the fingertips could encode contact onset, grip force, or material compliance by modulating vibration amplitude and frequency. Studies on surgical teleoperation show that such feedback can reduce inadvertent tissue damage by up to thirty percent [36]. Transferring those insights to industrial manipulation would require coupling the UR's internal force-torque observer to a high-level event classifier that triggers context-appropriate haptic codes.

Beyond these hardware-centric avenues lie algorithmic questions. Adaptive impedance control, in which the robot autonomously modulates stiffness based on inferred task intent, could be layered atop the direct mapping to produce a

shared-control regime. Embedding similar intelligence into a dual-arm framework would require coordinated impedance shaping to avoid internal stresses on a shared payload, a nontrivial optimization problem ripe for machine-learning approaches such as reinforcement learning with safety-constrained reward functions.

A related algorithmic opportunity lies in the safety and gesture-classification logic itself. The present system relies on fixed, deterministic thresholds: a 0.80 grab-strength crossing to trigger gripper closure, a 200 ms occlusion timeout, and a 500 ms double palm-down hold for the gesture halt, chosen empirically rather than derived from a probabilistic model. Replacing these hard cutoffs with a probabilistic rule-base, for instance, a Bayesian or hidden-Markov classifier that estimates a continuous confidence score for each gesture state from the Leap Motion's tracking-confidence and velocity signals, could reduce false triggers near the threshold boundary and provide a principled basis for the performance indices reported in Section 4. This was not implemented in the present work and is proposed here as a concrete extension to the otherwise deterministic safety architecture described in Section 3.4.

The article establishes a robust foundation for a feasible, affordable, and seamless method of controlling bimanual teleoperation systems. The observed metrics already exceed those reported in many mode-switching architectures, and the future pathways outlined here promise to deepen autonomy, enlarge the task envelope, and restore the rich haptic dialogue that supports true human dexterity.

Acknowledgment

The authors are grateful to faculty and staff of the University of New Haven for their unwavering support throughout the study.

Funding Support

This work is sponsored by the Department of Mechanical and Industrial Engineering, University of New Haven.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

Cheryl Q Li is an Editorial Board Member for *Journal of Climbing and Walking Robots* and was not involved in the editorial review or the decision to publish this article. The authors declare that they have no conflicts of interest in this work.

Data Availability Statement

Data are available from the corresponding author upon reasonable request.

Author Contribution Statement

Keerthivaasan Matheswaran: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data Curation, Writing – original draft, Visualization. **Cheryl Q Li:** Conceptualization, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

References

[1] Wang, T., Zheng, P., Li, S., & Wang, L. (2024). Multimodal human–robot interaction for human-centric smart manufacturing: A survey. *Advanced Intelligent Systems*, *6*, 2300359. https://doi.org/10.1002/aisy.202300359

[2] Singh, R., Mozaffari, S., Akhshik, M., Ahamed, M.J., Rondeau-Gagné, S., & Alirezaee, S. (2023). Human–robot interaction using learning from demonstrations and a wearable glove with multiple sensors. *Sensors*, *23*(24), 9780. https://doi.org/10.3390/s23249780

[3] Weichert, F., Bachmann, D., Rudak, B., & Fisseler, D. (2013). Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, *13*(5), 6380–6393. https://doi.org/10.3390/s130506380

[4] Basile, F., Caccavale, F., Chiacchio, P., Coppola, J., & Curatella, C. (2012). Task-oriented motion planning for multi-arm robotic systems. *Robotics and Computer-Integrated Manufacturing*, *28*(5), 569–582. https://doi.org/10.1016/j.rcim.2012.02.007

[5] Correia, A., & Alexandre, L. A. (2024). A survey of demonstration learning. *Robotics and Autonomous Systems*, *182*, 104812. https://doi.org/10.1016/j.robot.2024.104812

[6] Wang, L., Chen, Z., Han, S., Luo, Y., Li, X., & Liu, Y. (2025). An intuitive and efficient teleoperation human–robot interface based on a wearable myoelectric armband. *Biomimetics*, *10*(7), 464. https://doi.org/10.3390/biomimetics10070464

[7] Jeong, J., Choi, H., & Lee, D. (2025). Multi-mode hand gesture-based VR locomotion technique for intuitive telemanipulation viewpoint control in tightly arranged logistic environments. *Sensors*, *25*(4), 1181. https://doi.org/10.3390/s25041181

[8] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., . . . , & Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, *56*(1), 116–124. https://doi.org/10.1145/2398356.2398381

[9] Li, Y., Lou, J., Cai, Z., Zheng, P., Wu, H., & Wang, X. (2024). An interactive gesture control system for collaborative manipulator based on Leap Motion Controller. *Advances in Mechanical Engineering*, *16*(5), 1–18. https://doi.org/10.1177/16878132241253101

[10] Apostolellis, P., Bortz, B., Peng, M., Polys, N., & Hoegh, A. (2014). Poster: Exploring the integrality and separability of the Leap Motion Controller for direct manipulation 3D interaction. *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, 153–154. https://doi.org/10.1109/3DUI.2014.6798866,

[11] Hernoux, F., Béarée, R., & Gibaru, O. (2015). Investigation of dynamic 3D hand motion reproduction by a robot using a Leap Motion. In *Proceedings of the 2015 Virtual Reality International Conference*, 1–10. https://doi.org/10.1145/2806173.2806196

[12] Jin, H., Zhang, L., Rockel, S., Zhang, J., Hu, Y., & Zhang, J. (2015). A novel optical tracking based tele-control system for tabletop object manipulation tasks. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 636–642. https://doi.org/10.1109/IROS.2015.7353439

[13] Jin, H., Chen, Q., Chen, Z., Hu, Y., & Zhang, J. (2016). Multi-LeapMotion sensor based demonstration for robotic refine tabletop object manipulation task. *CAAI Transactions on Intelligence Technology*, *1*(1), 104–113. https://doi.org/10.1016/j.trit.2016.03.010

[14] Marin, G., Dominio, F., & Zanuttigh, P. (2014). Hand gesture recognition with leap motion and kinect devices. In *2014 IEEE International Conference on Image Processing (ICIP)*, 1565–1569. https://doi.org/10.1109/ICIP.2014.7025313

[15] Ghosh, S., Chatterjee, A., Munshi, S., & Rakshit, A. (2022). Flag-based human supervised robot guidance using locality preserving projections. In *2022 2nd International Conference on Emerging Frontiers in Electrical and Electronic Technologies (ICEFEET)*, 1–6. https://doi.org/10.1109/ICEFEET51821.2022.9848242

[16] Abbas, M., Narayan, J., & Dwivedy, S. K. (2023). A systematic review on cooperative dual-arm manipulators: Modeling, planning, control, and vision strategies. *International Journal of Intelligent Robotics and Applications*, *7*, 683–707. https://doi.org/10.1007/s41315-023-00292-0

[17] Lu, C., Jin, L., Liu, Y., Wang, J., & Li, W. (2024). Teleoperated grasping using data gloves based on fuzzy logic controller. *Biomimetics*, *9*(2), 116. https://doi.org/10.3390/biomimetics9020116

[18] Du, G., & Zhang, P. (2015). A markerless human–robot interface using particle filter and Kalman filter for dual robots. *IEEE Transactions on Industrial Electronics*, *62*(4), 2257–2264. https://doi.org/10.1109/TIE.2014.2362095

[19] Sosa-Ceron, A. D., Gonzalez-Hernandez, H. G., & Reyes-Avendaño, J. A. (2022). Learning from demonstrations in human–robot collaborative scenarios: A survey. *Robotics*, *11*(6), 126. https://doi.org/10.3390/robotics11060126

[20] Lu, Z., Wang, N., Li, Q., & Yang, C. (2023). A trajectory and force dual-incremental robot skill learning and generalization framework using improved dynamical movement primitives and adaptive neural network control. *Neurocomputing*, *521*, 146–159. https://doi.org/10.1016/j.neucom.2022.11.076

[21] Lenz, C., & Behnke, S. (2021). Bimanual telemanipulation with force and haptic feedback and predictive limit avoidance. In *2021 European Conference on Mobile Robots (ECMR)*, 1–7. https://doi.org/10.1109/ECMR50962.2021.9568842

[22] Fei, H., Xue, T., He, Y., Lin, S., Du, G., Guo, Y., & Wang, Z. (2025). Large language model-driven natural language interaction control framework for single-operator bimanual teleoperation. *Frontiers in Robotics and AI*, *12*, 1621033. https://doi.org/10.3389/frobt.2025.1621033

[23] Chicaiza, F. A., Slawiński, E., & Mut, V. (2025). Dual-arm mobile manipulator teleoperation with coupled rate-position mapping under time-varying delays. *IEEE Access*, *13*, 103463–103481. https://doi.org/10.1109/ACCESS.2025.3577009

[24] Wang, F., Chen, F., Dong, Y., Yong, Q., Yang, X., Zheng, L., . . . , & Su, H. (2023). Whole-body teleoperation control of dual-arm robot using sensor fusion. *Biomimetics*, *8*(8), 591. https://doi.org/10.3390/biomimetics8080591

[25] Ye, C., Yang, J., & Ding, H. (2022). Bagging for Gaussian mixture regression in robot learning from demonstration. *Journal of Intelligent Manufacturing*, *33*(3), 867–879. https://doi.org/10.1007/s10845-020-01686-8

[26] Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., & Schaal, S. (2013). Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, *25*(2), 328–373. https://doi.org/10.1162/NECO_a_00393

[27] Kamtam, S. B., Lu, Q., Bouali, F., Haas, O. C., & Birrell, S. (2024). Network latency in teleoperation of connected and autonomous vehicles: A review of trends, challenges, and mitigation strategies. *Sensors*, *24*(12), 3957. https://doi.org/10.3390/s24123957

[28] Ultraleap Ltd. (2020). "Leap Motion Controller™ Data Sheet, Issue 3."

[29] Universal Robots. (2020). "Real-Time Data Exchange (RTDE) Interface."

[30] Syed, I. S., Matheswaran, K., & Li, C. Q. (2024). Integrating mediapipe module for learn from demonstration on cobots. *International Symposium on Flexible Automation, 87882*, V001T07A013. https://doi.org/10.1115/ISFA2024-141320

[31] Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *13*(4), 376–380. https://doi.org/10.1109/34.88573

[32] Matheswaran, K., & Li, C. Q. (2025). Control of dual robotic arms via marker-less hand tracking. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 89251*, V005T08A037. https://doi.org/10.1115/DETC2025-169519

[33] Hartmann, D., Hamříková, K., Vysocký, A., Laciok, V., & Bernatík, A. (2026). Evolution of safety requirements in industrial robotics: Comparative analysis of ISO 10218-1/2 (2011 vs 2025) and integration of ISO/TS 15066. *Results in Engineering*, *30*, *110486*. https://doi.org/10.1016/j.rineng.2026.110486

[34] Matuszczyk, D., Jedrusiak, M., Fisseler, D., & Weichert, F. (2025). Comparative analysis of the accuracy and robustness of the leap motion controller 2. *Sensors*, *25*(24), 7473. https://doi.org/10.3390/s25247473

[35] Noguera Cundar, A., Fotouhi, R., Ochitwa, Z., & Obaid, H. (2023). Quantifying the effects of network latency for a teleoperated robot. *Sensors*, *23*(20), 8438. https://doi.org/10.3390/s23208438

[36] Raveh, E., Portnoy, S., & Friedman, J. (2018). Adding vibrotactile feedback to a myoelectric-controlled hand improves performance when online visual feedback is disturbed. *Human Movement Science*, *58*, 32–40. https://doi.org/10.1016/j.humov.2018.01.008

How to Cite: Matheswaran, K., & Li, C. Q. (2026). Real-Time Dual-Arm Teleoperation via Similarity-Transform Mapping. *Journal of Climbing and Walking Robots*. https://doi.org/10.47852/bonviewJCWR62029715