Robot Gesture Control using Online Feedback Data with Multi-Tracking Capture System

Khang H. V. Nguyen[1], Tri Bien Minh[2], Xuan Phu Do[1,*]

ABSTRACT
This study employs a Vicon, high-speed and accurate marker-based motion capture system to develop a wireless low-latency gesture control method that requires no additional hardware components or onboard power sources. A novel control method is designed to track a rigid-body object's motion in the workspace and translate it into robot control signals. The system comprises five main components: a host PC with Vicon software named Tracker, a set of eight Vicon Vantage V8 infrared cameras, a client PC that receives Tracker's broadcasted data over Ethernet, a robot model named youBot provided by KUKA, and a rigid-body object that acts as the gesture controller. By using an Application Programming Interface (API) called DataStream, the rigid-body object's pose (position and orientation) can be estimated and used to remotely control the KUKA youBot robot with higher accuracy and an overall reduced latency of 0.3 seconds. In comparison to existing methods based on accelerometer-based and gyroscope-based controllers, the proposed gesture controller uses a set of retro-reflective markers for tracking purposes and requires no additional hardware systems. This proposed control method proves especially beneficial for the robotics prototyping development phase, where different control mechanisms can be easily replicated and tested within the motion capture system's operating range. Additionally, the robot's odometry can also be tracked during testing, making a motion capture system a valuable tool for evaluating motion control methods.

Key words: Human-robot interaction, motion capture, wireless, gesture control, robotics, ROS

[1]*Mechatronics and Sensor Systems Technology, Vietnamese-German University, Binh Duong, Vietnam*

[2]*Electrical and Computer Engineering, Vietnamese-German University, Binh Duong, Vietnam*

Correspondence

Xuan Phu Do, Mechatronics and Sensor Systems Technology, Vietnamese-German University, Binh Duong, Vietnam

Email: phu.dx@vgu.edu.vn

History
- Received: 2023-08-04
- Accepted: 2023-09-20
- Published Online: 2023-09-30

DOI :
https://doi.org/10.32508/stdj.v26i3.4141

INTRODUCTION

In robotics development, particularly during the prototyping phase, troubleshooting tasks often require manual control to verify the actuator's responses. Developers can use various devices to input control signals, such as a keyboard, control panel, or joystick. Wireless devices are more favorable and commonly connect via Bluetooth. Although these devices are easy to use, they are primarily built with push buttons and potentiometers, thus limiting the input options. One of the new input methods is using gesture-related signals, for example, hand gestures detected by either using sensors (such as accelerometers [1,2], flex sensors [3], force sensors [4], or a combination of all these sensors [5]) or image processing. Due to the development of small handheld devices (smartphones), sensors are getting smaller and taking up less space on miniature circuit boards. Consequently, the sensors can easily be fitted on a glove [5] or a wristband [4], together with wireless units and batteries. The sensor output signals will then be programmed to control the robot's actuators; for instance, tilting the wrist (alters the gyroscope's output) controls steering, and bending the finger (alters the flex sensor's output) accelerates the robot. Additionally, gestures can be recognized using image processing techniques combined with machine learning methods to improve accuracy and reliability [6].

However, wireless sensor systems require an onboard power source and connectivity programming, while image processing techniques have relatively lower accuracy and require a proper dataset for feature matching and recognition [7]. This paper proposes a novel approach to robot gesture control that offers the following contributions in designing a gesture input method for robotics applications:

- Effortlessly mapping gestures to input for wireless robot control without building a dedicated unit that consists of an onboard power source, electronic components, sensors, and connectivity modules such as Bluetooth or Wi-Fi receivers.
- Supporting rapid experimentation of mechanical designs for gesture controllers.
- Providing an additional feedback signal for verifying the system's output by frequently tracking the target robot's motions.

By using a marker-based motion capture system, an object's pose (position and orientation in the three-dimensional (3D) coordinate system) can be tracked and processed to generate control signals to manipulate the robot's movement. Moreover, the tracked object has no fixed form or control mechanism; thus, developers can freely experiment with new developing control methods. The motion capture system used in this study is provided by Vicon, one of the most popular suppliers in the motion tracking solution market. By extracting motion capture data as the control signal, this study's gesture controller successfully controlled a KUKA youBot robot to move omnidirectionally with only one drawback: workspace restriction. The controller must remain within the capture range of the motion capture system to be effective. Therefore, while being simpler and easier to implement, the proposed controller is more useful during prototyping and testing phases rather than becoming an actual commercial product.

Related works

In addition to the marker-based motion capture system used in this study, 3D camera systems are often employed in human-robot interaction experiments. The most popular ones are the Kinect camera and Leap Motion controller[8]. By using the Leap Motion controller to track hand poses, one study developed a framework to remotely control a KUKA youBot robot arm and used a Head Mounted Display to monitor the system output by viewing the live recording of a Kinect camera mounted on the robot's end-effector[9]. However, the capture range of the Leap Motion controller is limited to 60 cm in front of the camera units, with a field of view of 140 degrees. Therefore, the Leap Motion controller is mostly suitable for tracking hand gestures. Conversely, the Kinect camera has a capture range of up to 4.5 m from the camera and is often used for tracking the whole body. By utilizing the Kinect camera and an intelligent decision-making method, another study proposed a framework where the system recognizes body gestures to command the robot to execute assigned tasks such as start, stop, and move up or down[10]. However, these body gestures are recognized from a predefined dataset instead of accurately tracking limbs in 3D. This study proposes a system where the gesture control unit can be freely moved in a larger workspace while its positions and orientations are accurately tracked in 3D, thus providing more options in gesture input method development.

MULTITRACKING ROBOT CONTROL SYSTEM

The proposed controller utilizes motion capture data to drive a KUKA youBot robot model. As illustrated in Figure 1, the system consists of eight calibrated Vicon Vantage V8 infrared cameras with a resolution of 8 megapixels (MP) and a frame rate set at 100 Hz[11]. Although the frame rate can be set to a higher value (up to 2 kHz), it is not recommended due to the significant decrease in the field of view. The resolution of 8 MP is sufficient to capture the images of the markers while excluding other objects visible in the capture range.

As also depicted in Figure 1, a rigid-body object equipped with retro-reflective markers serves as the gesture controller. By rotating this object about its axes, the gesture controller is able to drive the robot to move in various directions at velocities that are described in Figure 1. Furthermore, the technical details of the robot are discussed in greater depth in section II-B.

The devices are positioned in an experimental environment with dimensions of 4 m x 4 m x 2 m, as shown in Figure 2. This dimension is called the capture volume. The software Tracker processes the marker images captured by the Vantage cameras and broadcasts the data over Ethernet[12]. Additionally, Vicon provides a free API called DataStream, which enables client PCs in the same network to obtain capture data via Ethernet and develop third-party applications[13].

In this study, the proposed system captures the motion of the gesture input unit and translates it into control signals to drive the robot's movement. Furthermore, the robot's movement is also captured for verification. The robot is programmed to move at different linear and angular velocities, as described in Table 1, where a, b, and c are velocity factors. The details of the motion capture system, KUKA youBot, and data processing are discussed in more depth in the following subsections.

A. Vicon motion capture system

The motion capture system used in this study is marker-based and requires markers coated with retro-reflective material for tracking purposes. The target markers for the experiments have a diameter of 14 mm, making them suitable for small-scale robotics development and clinical analysis. Larger markers with a diameter of 25 mm are more beneficial for outdoor analyses, while smaller markers with a diameter of 9.5 mm are used to capture finer details such as facial expressions.

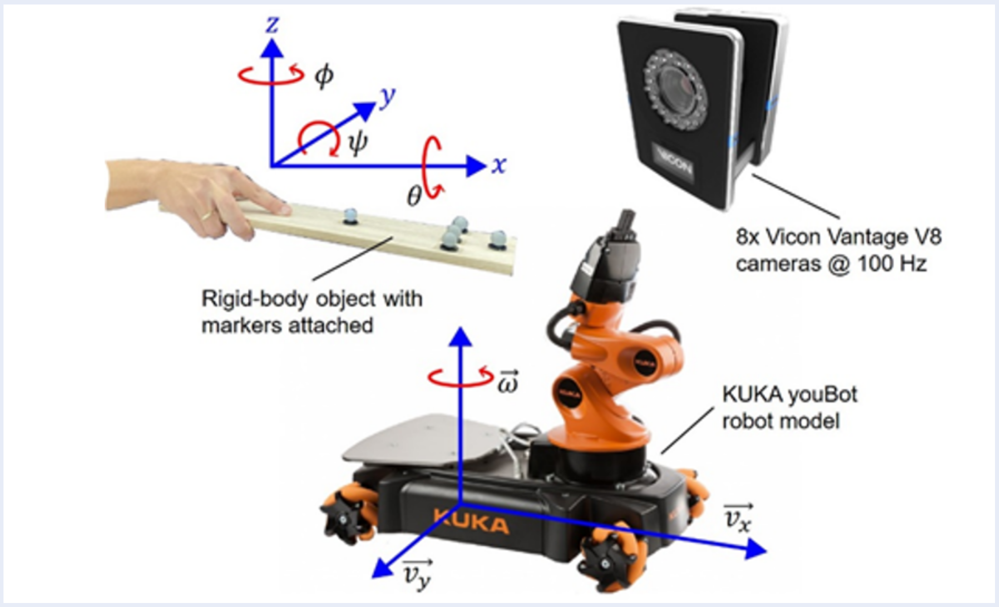

Figure 1: Experimental setup with the multitracking system and robot.

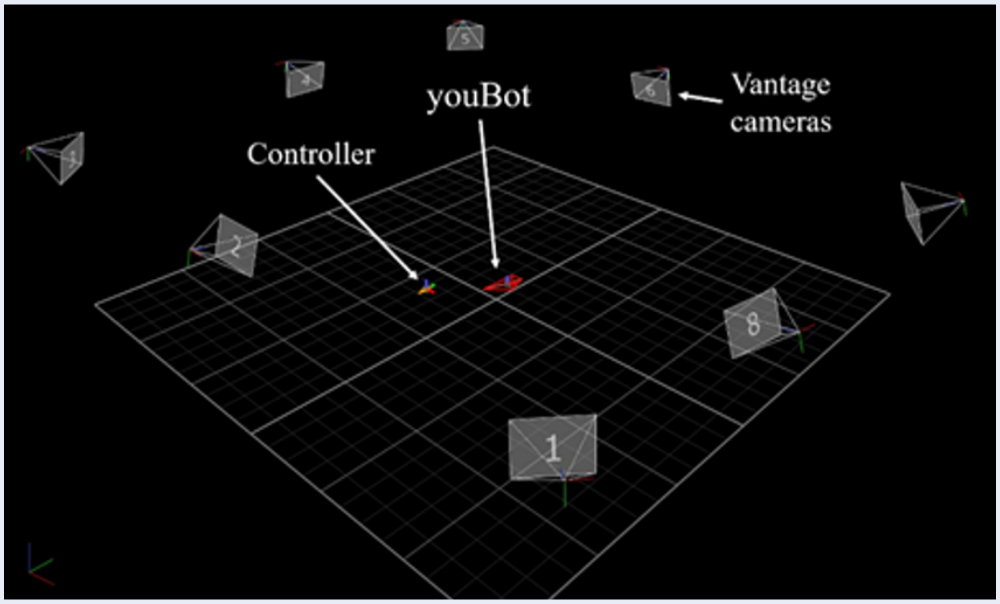

Figure 2: Camera positions and tracked objects in the capture volume.

Table 1: Robot velocities based on gesture input rotation angles

Rotation axes	Angles	Robot's velocities
x	θ	$v_y = \theta \times a\,(m/s)$
y	ψ	$v_x = \psi \times b\,(m/s)$
z	ϕ	$\omega = \phi \times c\,(m/s)$

To capture images that only include the markers, the Vantage cameras emit infrared light beams into the workspace using a strobe ring equipped on each unit. The cameras are adjusted to receive only the reflected light from the markers visible in the capture volume, as shown in Figure 3. When the markers are attached to objects, the centroid of each marker is estimated as a position M$_i$ in 3D (where i = 1, 2, 3, ..., n and n is the maximum number of markers visible in the capture volume).

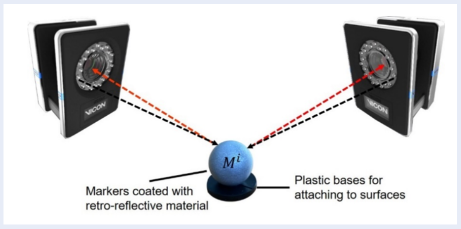
M^i
Markers coated with retro-reflective material
Plastic bases for attaching to surfaces

Figure 3: The marker reflects the light beam to the cameras.

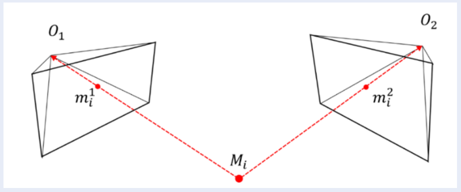

Figure 4: Projections of marker i in the camera image planes.

Figure 4 represents point M$_i$ as the position of marker i in 3D, and m_i^1 and m_i^2 are the corresponding 2D positions of marker i in the images captured by camera 1 and camera 2. O$_1$ and O$_2$ are the camera centers. The position of marker i in the 2D images can be computed as follows:

$$m_i^j = p^j M_i \tag{1}$$

where i is the target marker captured by camera j and P^j is the parameter matrix of camera j that will be generated during calibration.

$$m_i^j = w\begin{bmatrix} u_i^j & v_i^j & 1 \end{bmatrix}^T \tag{2}$$

$$p^j = \begin{bmatrix} p_{11}^j & p_{12}^j & p_{13}^j & p_{14}^j \\ p_{21}^j & p_{22}^j & p_{23}^j & p_{24}^j \\ p_{31}^j & p_{32}^j & p_{33}^j & p_{34}^j \end{bmatrix} = \begin{bmatrix} p_1^{jT} \\ p_2^{jT} \\ p_3^{jT} \end{bmatrix} \tag{3}$$

$$M_i = \begin{bmatrix} X_i & Y_i & Z_i & 1 \end{bmatrix}^T \tag{4}$$

where w is the scaling factor, u_i^j, v_i^j is the 2D position of marker i in the image captured by camera j and X_i, Y_i, Z_i is the marker's position from the 3D global coordinate origin. Thus,

$$wu_i^j = p_1^{jT} M_i \tag{5}$$

$$wv_i^j = p_2^{jT} M_i \tag{6}$$

$$w = p_3^{jT} M_i \tag{7}$$

By replacing the scaling factor w from equation $w = p_3^{jT} M_i$ in equations $wu_i^j = p_1^{jT} M_i$ and $wv_i^j = p_2^{jT} M_i$, the following equations can be obtained:

$$u_i^j p_3^{jT} M_i - p_1^{jT} M_i = 0 \tag{8}$$

$$v_i^j p_3^{jT} M_i - p_2^{jT} M_i = 0 \tag{9}$$

Thus, by capturing the images of point M$_i$ using a stereo camera setup (as shown in Figure 3), two sets of m_i^j and P^j can be acquired to form the following equation:

$$\begin{bmatrix} u_i^1 p_3^{1T} - p_1^{1T} \\ v_i^1 p_3^{1T} - p_2^{1T} \\ u_i^2 p_3^{2T} - p_1^{2T} \\ u_i^2 p_3^{2T} - p_1^{2T} \end{bmatrix} M_i = AM_i = 0 \tag{10}$$

Given:

$$A = \begin{bmatrix} u_i^1 p_{31}^1 - p_{11}^1 & u_i^1 p_{32}^1 - p_{12}^1 & u_i^1 p_{33}^1 - p_{13}^1 & u_i^1 p_{34}^1 - p_{14}^1 \\ v_i^1 p_{31}^1 - p_{21}^1 & v_i^1 p_{32}^1 - p_{22}^1 & v_i^1 p_{33}^1 - p_{23}^1 & v_i^1 p_{34}^1 - p_{24}^1 \\ u_i^2 p_{31}^2 - p_{11}^2 & u_i^2 p_{32}^2 - p_{12}^2 & u_i^2 p_{33}^2 - p_{13}^2 & u_i^2 p_{34}^2 - p_{14}^2 \\ v_i^2 p_{31}^2 - p_{21}^2 & v_i^2 p_{32}^2 - p_{22}^2 & v_i^2 p_{33}^2 - p_{23}^2 & v_i^2 p_{34}^2 - p_{24}^2 \end{bmatrix} \tag{11}$$

Consequently, using more than two cameras will add more sets of m_i^j and P^j to matrix A. Since M_i is the position of marker i in 3D, letting M$_i$ = 0 would be undesirable. Therefore, M$_i$ should be calculated in a way that minimizes ||AM$_i$|| subject to ||M$_i$||=1 [14]. Assume that:

$$A = UDV^T \tag{12}$$

where A is the matrix that contains the known values of marker positions in 2D m_i^j and camera matrices P^j, U is a complex unitary matrix, D is a rectangular diagonal matrix with nonnegative values on the diagonal and V is a complex unitary matrix. Using the singular value decomposition (SVD) method, the problem is deduced as follows:

$$M_i = V\begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^T \tag{13}$$

Therefore, the position of marker i in 3D can be estimated as the last column of matrix V, which is also the unit eigenvector of the matrix $A^T A$, where the smallest eigenvalue is determined using the SVD method. However, in practice, the two lines O_1M_i and O_2M_i may not intersect or intersect at another point in 3D that is not the marker's centroid[15]. Furthermore, using multiple cameras for improving 3D reconstruction may lead to noisy results, as multiple light beams are reflected and there are multiple intersections in 3D. One of the popular methods that is used after implementing reconstruction algorithms is bundle adjustment, which estimates the point in 3D while minimizing the reprojection error caused by multiple camera views:

$$\min_{P^j M_i} \Sigma_{ij} d\left(P^j M_i, m_i^j\right) \quad (14)$$

Figure 5 summarizes the overall process of estimating the marker position in 3D using the linear method discussed above, where matrix A is created using the camera projections and parameters, and the estimated position of the marker M_i in 3D is the unit eigenvector of $A^T A$; then, the last step is the minimization of the error between the measured and estimated positions using the bundle adjustment method[16]. After synchronizing the Vantage cameras and estimating the positions of all markers in the capture volume at one instance in time or frame, the motion capture system repeats the process for the next frame. The frame rate of the system is set at the default value of 100 Hz.

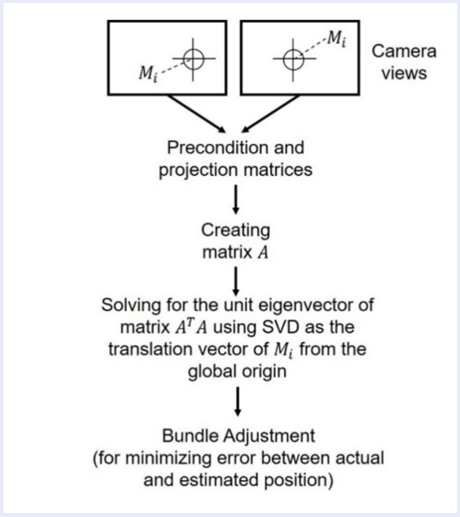
M_i
M_i
Camera views
Precondition and projection matrices
Creating matrix A
Solving for the unit eigenvector of matrix A^TA using SVD as the translation vector of M_i from the global origin
Bundle Adjustment (for minimizing error between actual and estimated position)

Figure 5: Linear method for 3D point estimation.

Although the frame rate of the cameras can be set to a higher value (up to 2 kHz), it is not recommended because it significantly decreases the capture volume. Prior to performing any measurements, the cameras must be calibrated to obtain the camera matrix P^j required for 3D reconstruction algorithms. A study conducted using eight Vicon T40S cameras found that the static system error was 0.02 mm and the dynamic system error was 0.2 mm[17]. Another study reported an accuracy of 63±5 μm in ideal conditions with four Vicon Mcam-60 cameras[18]. However, both Vicon T40S and Vicon Mcam-60 cameras are now obsolete. The cameras used in this study (Vantage V8) are still supported by Vicon and have been updated to the latest firmware version. Further details of the calibration procedure can be found on the official web page[11]. Once capturing begins, the tracked pose data can be processed locally or broadcast over Ethernet for client PCs to develop custom applications using the Vicon DataStream API. Although the API supports various programming languages, such as C, C++, dotNET, and Python[13], this study mainly implements robotics libraries written in Python due to the included libraries being written in this language. This suits the simplicity of the proposed model, where new control mechanisms can be quickly developed and experimented.

The pose data obtained using the DataStream API consist of translation vectors and rotation matrices of objects from the defined global origin of the workspace in each frame. To use these data for robot control, an additional step of spatial transformation is needed. These transformations will be discussed further in subsequent sections.

B. KUKA youBot robot system

As mentioned earlier, the target robot of this study is the KUKA youBot, a mobile robotic arm developed by KUKA for research and education. The robot features a 5-degree-of-freedom (5-DoF) arm and a two-finger linear gripper. Its base is equipped with four Mecanum wheels that enable omnidirectional movement, as shown in Figure 6. The robot also has an onboard PC board with an operating system, wireless module, and drivers installed, allowing it to execute stored programs or be remotely controlled by a computer[19]. Despite being announced as end-of-life, the youBot remains a solid platform for integrating third-party hardware and testing new motion control algorithms.

The robot's software can be developed using open-source robotics middleware such as Robot Operating System (ROS), Open Robot Control Software (Orocos), or the Robotics Module of LabVIEW. This paper focuses on developing the robot using ROS due to its widely available, community-supported packages for the youBot that have proven to be more reliable compared to other development platforms. ROS runs on the Linux operating system (OS), so a client PC with Ubuntu 16.04 installed is required to receive motion capture data broadcast over Ethernet and communicate wirelessly with the robot. The installed ROS version used in this study is Kinetic Kame, which is recommended for Ubuntu 16.04[20].

Linux is installed on a virtual machine rather than replacing the current OS or using dual-boot mode for convenience in backing up and restoring in case of system errors. For instance, restoring from a backup using a virtual machine is easier when an incompatible library has rendered scripts unusable.

The client PC used in this study is an Acer TravelMate P259-M with an Intel Core i5-6200U processor, 8 GB RAM, and a 500 GB solid-state disk drive. The main operating system is Microsoft Windows 10 Pro and Linux runs on a local virtual machine powered by VMWare Workstation 16 Player.

ROS communicates with the youBot by exchanging messages containing sensor data, motor control commands, state information, actuator commands, etc.[21]. As such, the motion capture data package from the Vicon system cannot be sent directly to the client PC and requires additional processing. In addition to installing ROS, a package called vicon_bridge must be included in the build system to translate messages received from Vicon Tracker into typical ROS message formats before sending commands to the KUKA youBot robot[22].

C. KUKA youBot mobile manipulator forward kinematics

Robotics arms are formed by a series of joints connected by links that show a kinematic chain. In a manipulator with an open kinematic chain, the total DoF is described by the total number of the robot's joints[23].

A joint variable q$_m$ is expressed as follows:

$$q_m = [q_1\ q_2\ q_3\ \cdots\ q_n]^T \quad (15)$$

where each joint value q_j with j = 1, 2, 3, 4, 5 represents an articulation and n =5 is the total DoF of the robotics KUKA youBot arm. The robot kinematics model can be collected based on its kinematics chain, as shown in Figure 6. Each link j is described by a homogeneous matrix ${}^{j-1}T_j$, which transforms the frame j-1 into the frame of link j. The kinematic model of the youBot can be expressed using the Denavit-Hartenberg (DH) convention[24]. The KUKA youBot DH table is shown in Table 2[25]. The homogeneous matrix ${}^{j-1}T_j$ can be expressed as:

$$ {}^{j-1}T_j = \begin{bmatrix} c(\vartheta_j) & -s(\vartheta_j)c(\beta_j) & s(\vartheta_j)s(\beta_j) & \alpha_j c(\vartheta_j) \\ s(\vartheta_j) & c(\vartheta_j)c(\beta_j) & -c(\vartheta_j)s(\beta_j) & \alpha_j s(\vartheta_j) \\ 0 & s(\beta_j) & c(\beta_j) & d_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1$$

where c and s represent the cos and sin operations, respectively. ϑ_j is the joint angle, d$_j$ is the link offset, α_j is the link length and β_j is the link twist.

The forward kinematics of the robot arm compute the end-effector pose ${}^0T_5$ given the joint variable q$_m$. This can be computed as:

$$ {}^0T_5(q_m) \overset{0}{=} T_1(q_1)^1 T_2(q_2)^2 T_3(q_3)^2 T_4(q_4)^2 T_5(q_5) = \prod_{j=1}^{n} {}^{j-1}T_j(q_j) \quad (17)$$

The matrix ${}^0T_5$ is described as:

$$ {}^0T_5 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \quad (18)$$

where the orientation and Cartesian position of the end-effector are represented by the matrix R and vector t, respectively.

In Figure 6, the world frame is denoted by the w frame with the (x$_w$, y$_w$, z$_w$) coordinate. The transformation ${}^wT_b$ represents a homogeneous matrix that expresses the orientation and position of the mobile youBot platform. The transformation ${}^bT_0$ represents a homogeneous matrix that coincides with the mobile robot base and the robot arm. The homogeneous matrix ${}^0T_5$ from frame 0 to frame 5 (end-effector frame) contains the position and orientation of the robot's end-effector.

The forward kinematics of the mobile manipulator are attainable as:

$$ {}^wTe(q) = {}^wT_5(q) = {}^wT_b(q_b)^b T_0 {}^0T_5(q_m) \quad (19)$$

where the matrix ${}^wTe(q)$ expresses the end effector pose with respect to the world coordinate frame w. The KUKA youBot is built on the omnidirectional mobile platform with Mecanum wheels, so the matrix ${}^wT_b$ represents the orientation ϑ_b rotating about the z-axis and positions x$_b$ and y$_b$ with respect to the world frame w of the robot. The matrix ${}^wT_b$ can be expressed as:

$$ {}^wT_b = \begin{bmatrix} \cos(\vartheta_b) & -\sin(\vartheta_b) & 0 & x_b \\ \sin(\vartheta_b) & \cos(\vartheta_b) & 0 & y_b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (20)$$

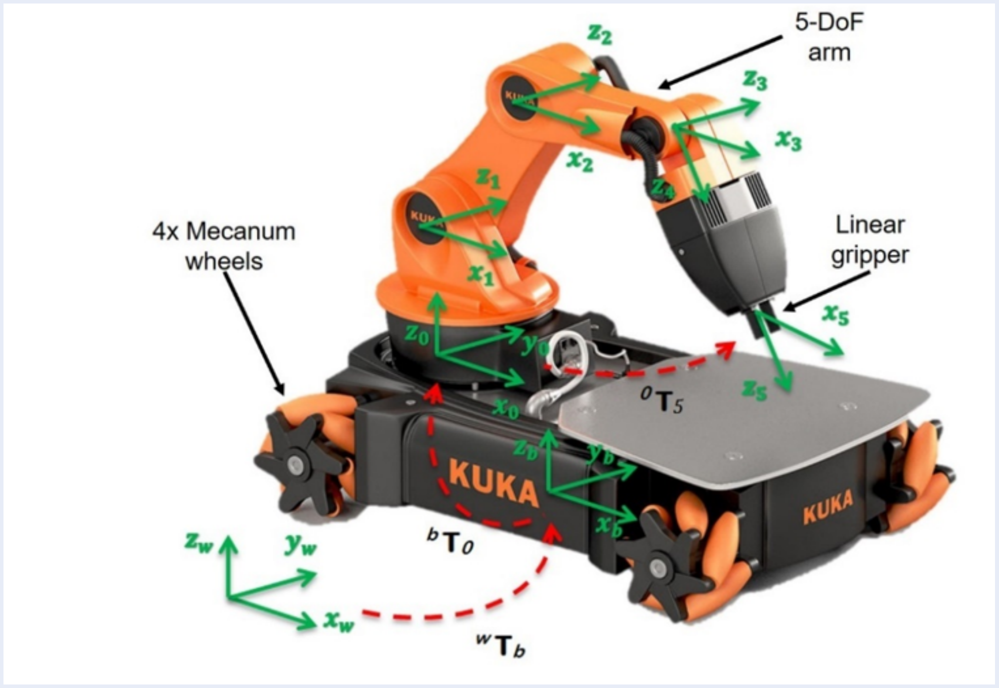

Figure 6: KUKA youBot mobile robotic arm

Table 2: DH Parameters Table for KUKA YouBot manipulator

Joints	(mm)	$\beta\,(rad)$	d (mm)	$\vartheta\ (rad)$
1	33.0		147.0	q_1
2	155.0	0.0	0.0	q_2
3	135.0	0.0	0.0	q_3
4	0.0		0.0	q_4
5	0.0	0.0	217.5	q_5

The matrix ${}^{b}T_0$ is constant, which includes the adjusted distance from the mobile base at frame b to the manipulator starting frame 0. The distances in the direction of the x-axis, y-axis and z-axis are t_x, t_y and t_z, respectively. The matrix ${}^{b}T_0$ can be defined as:

$$
{}^{b}T_0 = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}
$$

The joint variable of the mobile manipulator is described as follows:

$$
q = \left[q_b^T\ q_m^T\right]^T \tag{22}
$$

where the joint variable of the omnidirectional robot base configuration $q_b = [x_b\ y_b\ \vartheta_b]^T$ and the manipulator joint variable $q_m = [q_1\ q_2\ q_3\ q_4\ q_5]^T$.

D. Controller definition

As mentioned in the previous sections, this paper introduces a simple method for controlling a robot using a gesture controller, eliminating the need for complex controllers or multiple command lines. The controller is a wooden piece measuring 40 cm x 9 cm x 1 cm with five markers attached, as shown in Figure 7. In the Tracker interface, the markers are tracked and grouped as an object with its own local coordinate system. Any nonreflective and rigid object can be used, provided the markers are properly attached. The object's local coordinate system can be defined by the user or automatically set by Tracker. To improve orientation tracking accuracy, it is recommended to use four or five asymmetrically attached markers. Using the Vicon DataStream API, the transformation matrix of the tracked controller and its corresponding

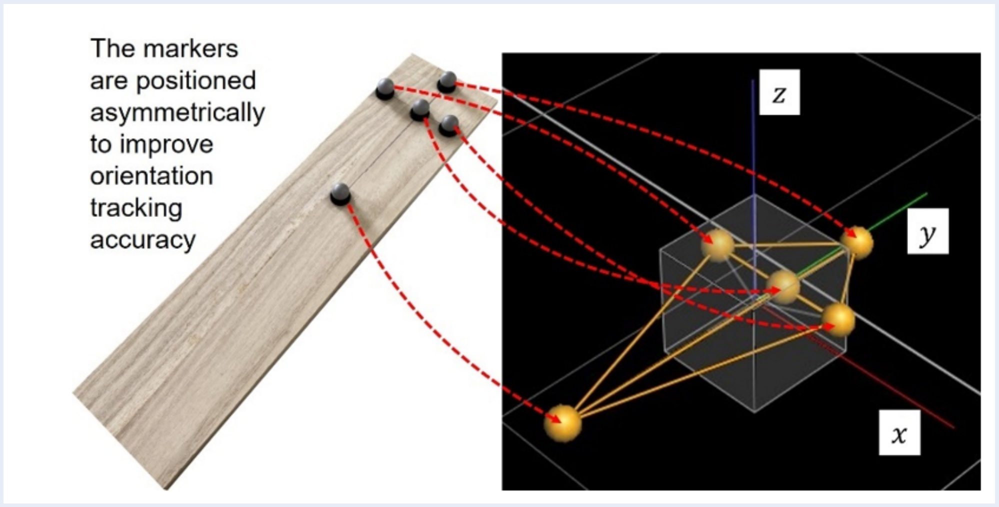
The markers are positioned asymmetrically to improve orientation tracking accuracy
z
y
x

Figure 7: The gesture controller and the respective representation in the tracker interface.

Euler angles are obtained and used as control parameters. To teleoperate the robot, the user rotates the controller about its axes, and the angle values determine the velocities as described in Table 2. This control mechanism is based purely on gestures, allowing for easy experimentation with different control schemes. For example, a plastic dish could be used as a steering wheel to drive a mobile robot. Additionally, chains of motion can be recorded as time series data to train a deep learning model, such as the Temporally Weighted Spatiotemporal Explainable Neural Network for Multivariate Time Series model[26], to recognize gestures and program the robot to react accordingly.

E. Controller kinematics

The Vicon DataStream API represents the pose of an object in each frame as a translation vector and rotation matrix. When an object i is detected in the workspace, its position and orientation in the global coordinate system are represented as follows:

$$t_i = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix} \tag{23}$$

$$R_i = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{24}$$

where t$_i$ is the translation vector of object i from the global coordinate and R$_i$ is the object's rotation matrix from the global coordinate. The transformation of an object i from the local coordinate to the global coordinate is:

$$^0T_i = \begin{bmatrix} R_i & (t_i)^T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{25}$$

However, it is more convenient to have the control signal only composed of the translation vector, and the rotation is described as the zyx sequenced three-angle representation, or Euler angles. According to Vicon[13], the rotation matrix is the product of the elementary rotation matrices about the x, y and z-axes:

$$R_i = R_x(\theta) R_y(\psi) R_z(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\theta) & -s(\theta) \\ 0 & s(\theta) & c(\theta) \end{bmatrix} \times \begin{bmatrix} c(\psi) & 0 & s(\psi) \\ 0 & 1 & 0 \\ -s(\psi) & 0 & c(\psi) \end{bmatrix} \begin{bmatrix} c(\phi) & -s(\phi) & 0 \\ s(\phi) & c(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{26}$$

where c and s represent cos and sin operations, respectively, while ψ, θ, and ϕ are the Euler angles. The rotation angle ϕ can be calculated from the obtained transformation matrix ^0 T$_i$ as:

$$\phi a \tan 2(r_{21}, r_{11}) \tag{27}$$

The rotation angles about the y- and x-axes can be determined as follows:

$$\theta = a\tan 2(-r_{31}, \cos(\phi) \times r_{11} + \sin(\phi) \times r_{21}) \tag{28}$$

$$\theta = a\tan 2(\sin(\phi) \times r_{31} - \cos(\phi) \times r_{23}, \cos(\phi) \times r_{22} - \sin(\phi) \times r_{21}) \tag{29}$$

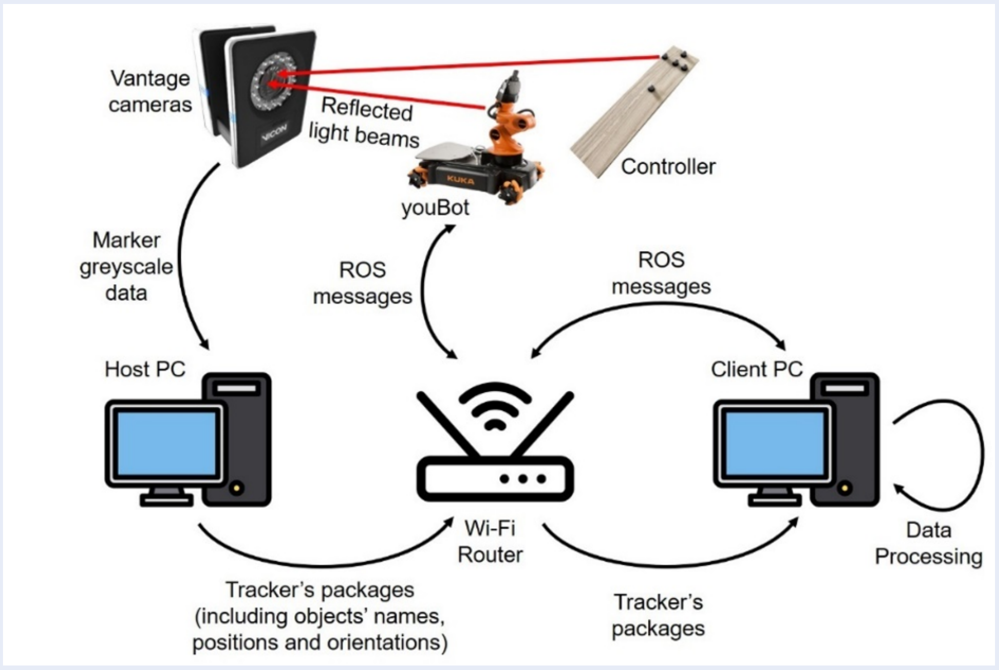

Figure 8: Gesture control system topology

F. System topology

The proposed controller utilizes motion capture data, as illustrated in Figure 8. Upon initiation of the system and visibility of the objects (the controller and the robot) within the capture volume, Vicon Tracker processes the marker images captured by the Vantage cameras. Subsequently, it broadcasts the data over Ethernet, including object names, translation vectors, and rotation matrices in the global coordinate system. As previously mentioned in section II-B, controlling youBot requires ROS middleware. A client PC with ROS installed receives the data package using the DataStream API. It then converts the rotation matrix into a three-angle representation called Euler angles and assigns these angles to control the linear and angular velocities of the robot. Since ROS communicates with youBot by exchanging messages, control messages consisting of gestures and corresponding actuator controls are transmitted from the client PC to youBot via Ethernet, causing youBot to move accordingly. It is important to note that as data transmission relies purely on Ethernet, all devices, including the host PC, client PC, and youBot, must be connected to the same network. Additionally, for proper communication, the IP address of each device must be specified in the source code.

RESULTS

To evaluate the performance of the proposed controller, the robot's base was equipped with five additional markers and tracked by the Vicon system along with the gesture controller described in section II-D, as shown in Figure 2. The robot was programmed to move according to the velocities listed in Table 3, which were calculated from the controller's angles as follows:

The recorded datasets were processed and plotted using MATLAB (version R2017a) on the client PC, as specified in section II-B. Unlike other methods that require filtering to reduce the noise in the measurements[27], the method of this study does not apply any filtering to the data to keep the experimental results as transparent as possible and avoid adding additional errors to the validation. This is because the Vicon system uses infrared cameras to track the markers attached to the objects and calculates their positions and orientations with submillimeter accuracy and precision. Therefore, proper camera calibration and positioning prove to be more important[17]. The linear velocities along the x and y-axes were derived from the position values of the global origin using Euler angles, as shown below:

$$v_x[n] = \frac{t_x[n] - t_x[n-1]}{0.01} \quad (30)$$

Table 3: youBot's velocities calculated from controller's angles

$v_x\,(m/s)$	$v_y\,(m/s)$	$\omega\,(rad/s)$
$0.6*\psi$	$0.6*\theta$	$0.4*\phi$

$$v_y[n] = \frac{t_y[n]-t_y[n-1]}{0.01} \quad (31)$$

where v$_x$ and v$_y$ represent the linear velocities along the x and y-axes, respectively. They were obtained by dividing the difference in positions by an interval of 0.01 s due to the Vicon system's 100 Hz sampling rate. Similarly, the angular velocity (e.g., rotation about the z-axis) was derived from the orientation values using the following equation:

$$\omega[n] = \frac{\phi youBot[n]-\phi youBot[n-1]}{0.01} \quad (32)$$

where ω represents the angular velocity and ϕ_{youBot} represents the orientation angle of the youBot robot. The performance of the proposed controller was evaluated by analyzing the experimental results. The results include the plots of the velocities, positions, and orientations of the robot and the controller, which are derived from the Vicon system data. The response time and reliability of the system are also discussed. The response time was defined as the time elapsed between the user's gesture input and the robot's motion output.

A. Performance

Figure 9 provides a detailed illustration of the comparison between the motion of the controller and the movement of the robot. The black line in the figure represents the angle values of the controller in 3D space, while the red line represents the calculated linear and angular velocities of the KUKA youBot robot. To simplify the visualization of the test results, the robot was controlled to move along only one direction at a time, avoiding any diagonal movements.

Overall, the controller is capable of effectively controlling the robot to move as it was programmed to do. The velocities of the robot are equal to the multiplications of the velocity factors and their corresponding Euler angles. However, as seen in Figure 9, there are some factors that affected the velocity response of the robot, such as its limited acceleration and deceleration capability and its base orientation.

One of these factors is the limited acceleration and deceleration capability of the robot, which prevents it from changing its speed instantly when the controller's angle changes abruptly. This caused a delay in the velocity response, which can be observed when changing direction. For example, around the 23rd second in Figure 9, when the controller's angle ψ changes from negative to positive, indicating a change in direction along the x-axis, there was a noticeable delay in the robot's velocity v$_x$ before it stabilizes at the desired value. This long settling time is due to the use of the default kinematic configuration of the Mecanum wheels, as provided in the youBot API[28]. This phenomenon is not exclusive to changes along the x-axis but is also observed when altering directions at various angles. To enhance the response time and minimize steady-state errors during high-speed movement control via the gesture controller, future experiments will incorporate advanced control methods. One such method under consideration is the self-tuning fuzzy PID-nonsingular fast terminal sliding mode control method[29].

Another factor that affected the velocity response of the robot is its base orientation, which also influences its linear velocities along the x and y-axes. When the robot rotates about the z-axis, its base frame changes relative to the global frame, and thus, its velocities in the global frame also change. This can be seen around the 50th second in Figure 9, where the robot's angular velocity ω is not zero but its linear velocities v$_x$ and v$_y$ are not proportional to the controller's angles ψ and θ. This discrepancy can be attributed to rotation about the z-axis, as shown in Figure 9's third graph. The displacement of the robot's base consequently changes its velocities along both its x and y-axes.

These two factors explain why the robot's velocity curve in Figure 9 is not smooth. However, they do not affect the user experience, as they still show a relatively satisfactory performance. Therefore, these reasons are not errors or flaws of the system but rather natural consequences of controlling a rigid body in 3D space with a gesture controller.

It is also important to note that as mentioned in section II-A, the accuracy of the motion capture system can be in a range of $\pm 100\,\mu$m. As such, minor shaking while holding the gesture controller may cause unintentional movement of the robot. Additionally, leaving the controller unnoticed while the system is still online may result in unpredictable damage. To avoid any unwanted accidents or incidents, after data have been recorded, the system is switched to offline mode, and all motors on the robot are disabled.

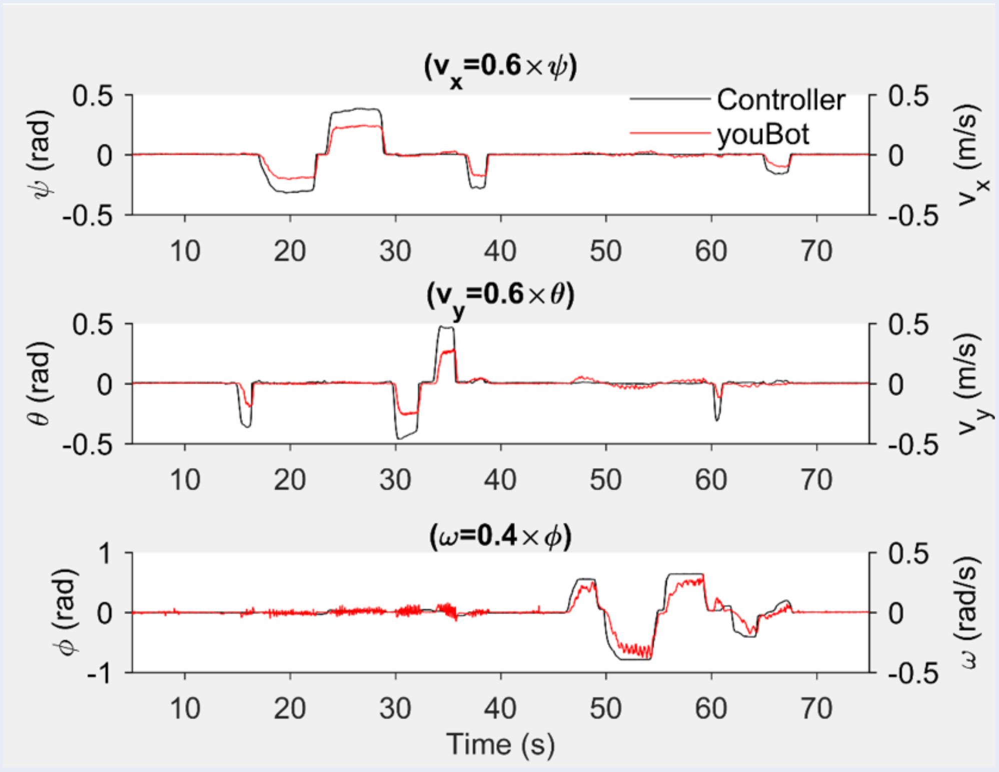

Figure 9: Comparison between controller's angles and robot's velocities.

B. Overall latency

Upon close inspection of the measurements taken over a 4-second duration, it is evident that the system's overall latency is approximately 0.3 seconds, as depicted in Figure 10. While this level of latency may not be suitable for hard real-time applications where delays can have fatal consequences, it is worth noting that the proposed system still provides an acceptable user experience. This can be attributed to the fact that human perception typically has a reaction time ranging between 0.2 and 0.3 seconds[29]. It is also important to note that the robot operating system (ROS) is not currently capable of delivering hard real-time performance. Meanwhile, ROS 2 is under development with the aim of improving real-time performance, although it still depends on various factors, such as computer architecture, operating system, and specific use case[30].

CONCLUSION

This study introduced a novel experimental setup for controlling the KUKA youBot robot system using a simple rigid-body object as a gesture input method. The object's movements were accurately tracked by the Vicon marker-based motion capture system, and the tracked pose data were processed into command messages and transmitted over Ethernet to drive the robot omnidirectionally. Evaluation showed that the proposed controller effectively controlled the target robot with an overall latency of 0.3 seconds. The proposed system has the potential to serve as a platform for experimenting with different motion-based control methods without requiring detailed hardware preparation. This means that developers can quickly prototype and research different control methods or gesture recognition algorithms. The system's simplicity and flexibility make it an attractive option for researchers and developers looking to explore new approaches to robot control. While the gesture controller is currently limited to a defined capture volume, data transmission over Ethernet can increase the working range of the system as long as the robot's odometry is properly monitored. This opens up new possibilities for controlling robots in larger environments and over greater distances. In future studies, we plan to investigate the proposed system's performance when driving the robot beyond the current capture volume. This will provide valuable insights into the system's capabilities and limitations and help guide

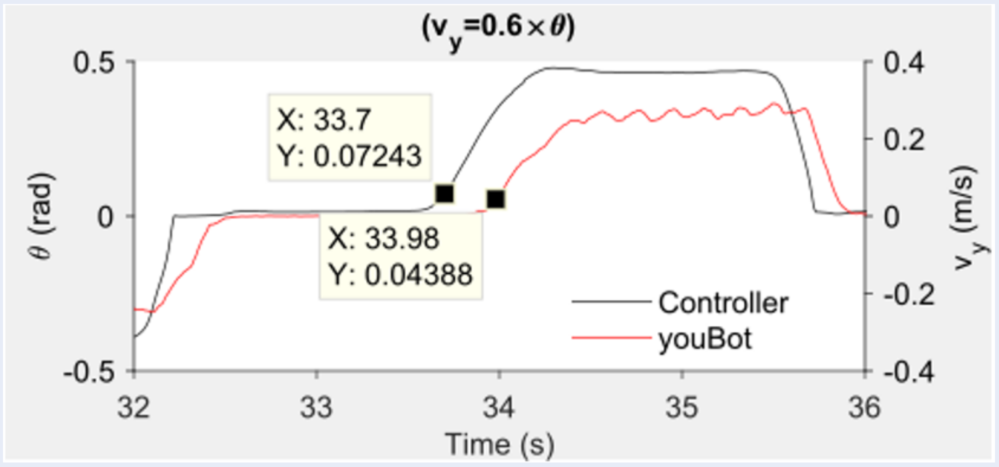

Figure 10: Typical system latency.

further development and refinement of the gesture-based control method.

ABBREVIATIONS

MP: Megapixels
SVD: Singular Value Decomposition
API: application programming interface
PC: Personal Computer
ROS: Robot Operating System
DoF: Degree of Freedom
DH: Denavit-Hartenberg
IP: Internet Protocol
PID: Proportional Integral Derivative

ACKNOWLEDGMENTS

This research is funded by the Ministry of Education and Training (Vietnam) under grant number B2024-VGU-02.

COMPETING INTERESTS

The authors declare that there are no conflicts of interest.

AUTHOR'S CONTRIBUTIONS

All authors significantly contributed to this work. All authors read and approved the final manuscript.

REFERENCES

1. Anderez DO, Dos Santos LP, Lotfi A, Yahaya SW. Accelerometer-based Hand Gesture Recognition for Human-Robot Interaction. 2019 IEEE Symposium Series on Computational Intelligence (SSCI). 2019;: 1402-1406;Available from: https://doi.org/10.1109/SSCI44817.2019.9003136.
2. Iyer P, Tarekar S, Dixit S. Hand Gesture Controlled Robot. 2019 9th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-19). 2019;: 1-5;Available from: https://doi.org/10.1109/ICETET-SIP-1946815.2019.9092032.
3. Panda AK, Chakravarty R, Moulik S. Hand Gesture Recognition using Flex Sensor and Machine Learning Algorithms. 2020 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES). 2021;: 449-453;Available from: https://doi.org/10.1109/IECBES48179.2021.9398789.
4. Anvaripour M, Saif M. Hand gesture recognition using force myography of the forearm activities and optimized features. 2018 IEEE International Conference on Industrial Technology (ICIT). 2018;: 187-192;Available from: https://doi.org/10.1109/ICIT.2018.8352174.
5. Dekate A, Kamal A, Surekha KS. Magic Glove - wireless hand gesture hardware controller. 2014 International Conference on Electronics and Communication Systems (ICECS). 2014;: 1-4;Available from: https://doi.org/10.1109/ECS.2014.6892518.
6. Raheja JL, Shyam R, Kumar U, Prasad PB. Real-Time Robotic Hand Control Using Hand Gestures. In Proceedings of the 2010 Second International Conference on Machine Learning and Computing; 2010; USA: IEEE Computer Society. p. 12-16;Available from: https://doi.org/10.1109/ICMLC.2010.12.
7. Oudah M, Al-Naji A, Chahl J. Hand Gesture Recognition Based on Computer Vision: A Review of Techniques. Journal of Imaging. 2020; 73;Available from: https://doi.org/10.3390/jimaging6080073.
8. Guzsvinecz T, Szucs V, Sik-Lanyi C. Suitability of the Kinect Sensor and Leap Motion Controller-A Literature Review. Sensors. 2019; 19(5): 1072;Available from: https://doi.org/10.3390/s19051072.
9. Peppoloni L, Brizzi F, Avizzano CA, Ruffaldi E. Immersive ROS-integrated framework for robot teleoperation. In 2015 IEEE Symposium on 3D User Interfaces (3DUI); 2015; Arles. p. 177-178;Available from: https://doi.org/10.1109/3DUI.2015.7131758.
10. Tsarouchi P, Matthaiakis AS, Makris S, Chryssolouris G. On a human-robot collaboration in an assembly cell. International Journal of Computer Integrated Manufacturing. 2017; 30(6): 580-589;Available from: https://doi.org/10.1080/0951192X.2016.1187297.
11. Vicon. [Online]. [cited 2022 09 19];Available from: https://docs.vicon.com/display/Vantage/Vicon+Vantage+camera+range.
12. Vicon. [Online]. [cited 2022 09 19];Available from: https://www.vicon.com/software/tracker/.
13. Vicon. [Online]. [cited 2022 09 19];Available from: https://docs.vicon.com/display/DSSDK111.
14. Hartley R, Sturm P. Triangulation. Computer Vision and Image Understanding. 1997; 68(2): 146-157;Available from: https://doi.org/10.1006/cviu.1997.0547.

15. Chen J, Wu D, Song P, Deng F, He Y, Pang S. Multi-View Triangulation: Systematic Comparison and an Improved Method. IEEE Access. 2020; 8: 21017-21027;Available from: https://doi.org/10.1109/ACCESS.2020.2969082.
16. Hartley R, Zisserman A. Multiple View Geometry in Computer Vision Cambridge: Cambridge University Press; 2004;Available from: https://doi.org/10.1017/CBO9780511811685.
17. Merriaux P, Dupuis Y, Boutteau R, Vasseur P, Savatier X. A Study of Vicon System Positioning Performance. Sensors. 2017;: 1591;Available from: https://doi.org/10.3390/s17071591.
18. Windolf M, Götzen N, Morlock M. Systematic accuracy and precision analysis of video motion capturing systems - exemplified on the Vicon-460 system. Journal of Biomechanics. 2008;: 2776-2780;Available from: https://doi.org/10.1016/j.jbiomech.2008.06.024.
19. youBot-Store. [Online]. [cited 2022 09 19];Available from: http://ftp.youbot-store.com/manuals/KUKA-youBot_UserManual.pdf.
20. Joseph L. Getting Started with Ubuntu Linux for Robotics Berkeley: Apress; 2018;Available from: https://doi.org/10.1007/978-1-4842-3405-1_1.
21. Mahtani A, Sanchez L, Fernandez E, Martinez A. Effective Robotics Programming with ROS - Third Edition: Packt Publishing; 2016;.
22. Achtelik M. vicon_bridge ROS package. [Online].; 2021 [cited 2022 09 19;Available from: https://github.com/ethz-asl/vicon_bridge.
23. Siciliano B, Sciavicco L, Villani L, Oriolo G. Robotics: Modeling, Planning and Control: Springer Publishing Company, Incorporated; 2008;.
24. Hartenberg RS, Denavit J. Kinematic synthesis of linkages. In McGraw-Hill series in mechanical engineering. New York: McGraw-Hill; 1965. p. 435;.
25. Hernandez-Barragan J, Lopez-Franco C, Arana-Daniel N, Alanis A. Inverse kinematics for cooperative mobile manipulators based on self-adaptive differential evolution. PeerJ Computer Science. 2021;: 419;Available from: https://doi.org/10.7717/peerj-cs.419.
26. Pham AD, Kuestenmacher A, Ploeger PG. TSEM: Temporally Weighted Spatiotemporal Explainable Neural Network for Multivariate Time Series. In Arai K, editor. Advances in Information and Communication.: Springer Nature Switzerland; 2023. p. 183-204;Available from: https://doi.org/10.1007/978-3-031-28073-3_13.
27. Skurowski P, Pawlyta M. On the Noise Complexity in an Optical Motion Capture Facility. Sensors. 2019; 19(20);Available from: https://doi.org/10.3390/s19204435.
28. Van M, Do XP, Mavrovouniotis M. Self-tuning fuzzy PID-nonsingular fast terminal sliding mode control for robust fault tolerant control of robot manipulators. ISA Transactions. 2020; 96: 60-68;Available from: https://doi.org/10.1016/j.isatra.2019.06.017.
29. Jensen AR. Chapter 3 - Reaction Time as a Function of Experimental Conditions. In Jensen AR. Clocking the Mind. Oxford: Elsevier Science Ltd; 2006. p. 43-54;Available from: https://doi.org/10.1016/B978-008044939-5/50004-5.
30. Yang Y, Azumi T. Exploring Real-Time Executor on ROS 2. 2020 IEEE International Conference on Embedded Software and Systems (ICESS). 2020;: 1-8;Available from: https://doi.org/10.1109/ICESS49830.2020.9301530.