A Virtual Mecanum Wheeled Robot ROS Simulator for Multi-view and Self-Following Motion Capture

Le Zhou, Nate Lannan, Cale England, and Guoliang Fan*

School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, USA
Email: le.zhou@okstate.edu (L.Z.); nate.lannan@okstate.edu (N.L.); cale.england@okstate.edu (C.E.); guoliang.fan@okstate.edu (G.F.)
*Corresponding author

***Abstract*—Motion capture (Mocap) on the go based on a mobile platform is valuable for clinical studies and rehabilitation. For multi-view gait analysis, Mecanum wheeled robots offer advantages over traditional differential drive robots. However, control issues in multi-view human tracking using Mecanum robots remain unexplored and lack a suitable virtual environment. This paper introduces a virtual Robot Operating System (ROS) environment with a Gazebo simulator as a research tool for multi-view human tracking on a Mecanum wheeled robot. The simulation incorporates a Proportional–Integral–Derivative (PID) controller and Kalman filter to maintain expected positional distance and relative viewing angles to the target. Our case study presents a quantitative evaluation of results obtained from the virtual environment for two specific tracking modes on a Mecanum wheeled robot: back-view following and side-view following with and without Kalman filtering. By optimizing the system, we decreased the distance error in backward following from 0.22 m to 0.12 m, and the angle error from 12.2° to 5.3°. Similarly, for side following, the distance error decreased from 0.32 m to 0.14 m, and the angle error reduced from 13.4° to 6.2°. These experimental results demonstrate that our approach enhances the accuracy of both tracking methods by over 50%. This work provides a necessary steppingstone for the development of human-tracking Mecanum wheeled robots for use in a clinical setting, providing a virtual environment for algorithmic development thereby eliminating wear on the hardware to be used in the clinical setting.**

***Keywords*—Robot Operating System (ROS), Proportional–Integral–Derivative (PID) control, multi-view human tracking, Mecanum wheeled robot, Kalman filter, robot control**

I. Introduction

Advancements in robotics have revolutionized various industries, and the field of biomedicine is no exception. One intriguing application is the development of human-following robots, designed to autonomously track and assist individuals within healthcare settings. This emerging technology holds tremendous promise, offering the potential to enhance patient care, streamline medical processes, and improve the overall efficiency of healthcare facilities. By seamlessly integrating robotics into the biomedical field, human-following robots have the capacity to augment medical professionals' capabilities, improve patient safety, and alleviate the strain on healthcare systems. Currently, many human-following robots in the biomedical field are assistive in nature, supporting human mobility and well-being [1]. Others are used in data collection applications such as, gait assessment [2–6]. In this paper, we explore the use of a Robot Operating System (ROS) based simulation environment for an omni-drive robot, with the aim at optimizing robotic vision algorithms that are intended on not only following a human subject from any perspective but also tracking the person and generating refined full-body 3D human motion kinematic data from the on-board RGB-D sensor.

While human-following robots offer promising potential in the biomedical field, their implementation is not without challenges. One of the main problems is mapping and navigation in a dynamic and unpredictable healthcare environment, where sudden changes in direction and obstacles hinder efficient robot movement. Additionally, the need for real-time tracking and accurate sensing technologies adds complexity to the system, requiring robust algorithms and sensor fusion techniques. Current methodologies cover a wide range of research including deep learning with Proportional–Integral–Derivative (PID) control [7], SURF-based tracking with Kalman filtering [8], lead-lag and PID control [3, 9], potential field algorithms [10], and Timed-Elastic-Band planning [4]. However, developing and testing these algorithms on physical robotic systems can be time-consuming and prone to maintenance delays. To overcome these limitations, researchers have turned to ROS-based simulation environments to refine human tracking algorithms before deploying them on hardware [11–13]. However, a drawback of current simulation environments is the lack of omnidrive models, as they typically use

Manuscript received February 3, 2024; revised April 8, 2024; accepted May 8, 2024; published October 8, 2024.

doi: 10.18178/ijmerr.13.5.522-529

differential drive locomotion. The problem with differential drive robotic systems is their lack of the ability to perform translation and rotation in any direction [14]. Using an omnidrive platform often simplifies the RGB-D mounting system for human pose estimation, eliminating the need for a dynamically controllable gimble. This eliminates the need to model the camera image in a separate frame as the robot thereby drastically simplifying practical application.

An omnidrive robot ROS simulation environment offers numerous advantages in the field of robotics research and development. One key advantage is the ability to simulate and test the performance of omnidrive robots in a virtual environment before deploying them in the real world. This simulation environment allows researchers and engineers to assess the robot's navigation capabilities, evaluate different control algorithms, and optimize its overall performance. By providing a realistic representation of the physical environment, including dynamic obstacles and varying terrain, the simulation environment enables researchers to anticipate and address potential challenges and limitations of the omnidrive robot. Additionally, the virtual environment allows for rapid prototyping and iterative design, facilitating quick iterations and modifications to improve the robot's efficiency and effectiveness. Furthermore, the simulation environment offers a cost-effective solution, as it eliminates the need for physical hardware and reduces the risks associated with testing in real-world scenarios. Overall, a Mecanum wheeled robot simulation environment provides a valuable platform for experimentation, innovation, and refinement, accelerating the development of robust and reliable Mecanum wheeled robot systems. All code used for this work will be made publicly available in the near future.

II. Literature Review

Related works primarily fall into two distinct categories: ROS simulation environments and human following control algorithms. By analyzing and categorizing the available literature into these two core areas, researchers can gain a comprehensive understanding of the current state-of-the-art, identify research gaps, and pave the way for future advancements in omnidrive human-following robots within the biomedical domain.

ROS-based simulation environments have emerged as a powerful tool for the development and evaluation of robotic systems, including human-following robots. There is a method for generating simulation environments in ROS and Gazebo with the aim of developing guiding robots for the elderly [12]. This paper acts as a practical tutorial for building a 2D or 3D simulation environment with the aim of designing a control algorithm for human-guiding robots. Similarly, another method uses ROS and Gazebo to build a simulation environment for a human-following robot [11]. This simulation environment also uses MakeHuman and Blender to build a more realistic environment in which depth RGB-D sensors can be simulated to track the human subject. There is another simulation environment for human-following [13] which provides a toolset for multi-robot tasks and can be easily extended to a leader-follower relationship to simulate a human subject and a following robot. The one aspect that all of these simulation environments lack is omnidrive robot control. To take advantage of the dynamic movement of an omnidrive system, we need a simulation environment designed for such a robot.

The current state of the art in human-following robots showcases a wide range of methodologies. A Single Shot Detector is used to identify the subject and then a PID controller is used to follow the subject once identified [7]. A SURF-based method was proposed to track a human subject while maintaining a desired position to that subject through a linear control system using Kalman filter velocities derived from the pinhole camera model [8]. Traditional lead-lag and PID controllers are used to maintain the robot follower's position with respect to the human subject [3, 9]. In contrast, a potential field algorithm is used to maintain the relative distance between the human and robot [10], and Elastic-Band planning was proposed that is more concerned with object avoidance in addition to the following of a human subject [4]. For all these methods, the vision system of the robot is overly complicated to compensate for the fact that differential drive robots lack the ability to perform smooth angular motion. While omnidrive control and navigation is a well-explored subject [5], the incorporation of the omnidrive into a human-following robot simulation environment is a contribution that will benefit the robotics community, especially regarding biomedical applications.

III. Mecanum-Wheeled Rbot Simulator

A well-scripted methods sections lays the foundation for your research by outlining the different methods you used to derive your results. The methods used to achieve the objectives must be described precisely and in sufficient detail, so as to allow a competent reader to repeat the work done by the author.

A. Introduction to the Basic ROS Nodes

Robot Operating System (ROS) is a flexible framework for writing robot software. One of its fundamental features is facilitating communication between different software modules called "nodes." This communication is essential for coordinating various tasks and sensors in a robotic system. The following is an introduction to the basic communication between ROS nodes:

(1) **Node:** In ROS, a "node" is a stand-alone software module that performs a specific task or function. Nodes are the building blocks of a ROS-based system. Each node can be written in various programming languages such as Python or C++, and they communicate with each other to achieve the desired robotic functionality.

(2) **Publisher-Subscriber:** ROS communication primarily follows the publisher-subscriber model. In this model, nodes communicate by publishing data on specific topics, and other nodes subscribe to those topics to receive and process the data. Topics act as channels for data exchange.

(3) **Topics:** A "topic" is a named bus over which data is exchanged between nodes. Topics are used to send and receive messages, which can be any type of data, such as sensor readings, control commands, or status updates. Nodes can publish data on one or more topics, and they can also subscribe to one or more topics.

(4) **Messages:** Messages are structured data types that define the format and content of data exchanged between nodes on a topic. ROS provides a set of predefined message types, and users can also define custom message types to suit their specific needs. For example, a laser range finder node might publish messages containing distance and angle information.

The schematic representation of the software architecture of the Mecanum wheeled robot simulator utilized in this experiment is illustrated in Fig. 1. The name of the node is documented within the elliptical shape, while the node from which the arrow originates assumes the role of the information publisher. Conversely, the node that the arrow points to assumes the role of the information subscriber. The arrow connecting line in the figure serves as a representation of the information conveyed in the Robot Operating System (ROS). This information is encapsulated within a topic, which contains more detailed digital data.

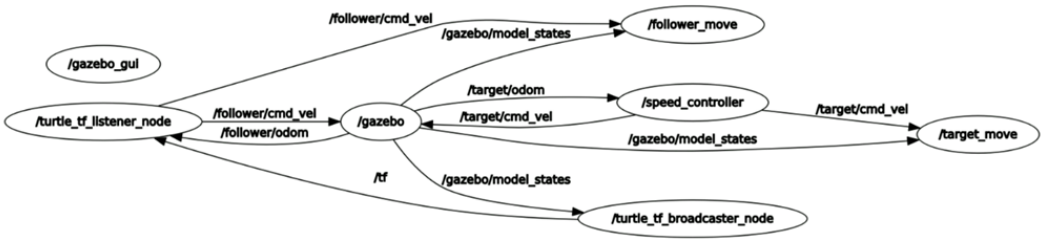

Fig 1. ROS gazebo simulator software framework for Mecanum robot simulation under ROS and Gazebo.

B. Robot Simulation under ROS and Gazebo

ROS models robots and their environs using the Universal Robotics Description Format (URDF) which is an Extensible Markup Language (XML) file format that has been established for representing robot structure. However, it is not a "universal" description format because some information necessary for other robotics domains cannot be depicted in URDF, such as specific joint loops, friction, and other sensor properties. Files written in the XACRO (XML Macros) format allow for the definition of specific attributes that are absent from URDF. For example, <commandTopic> and <odometryTopic> can be defined in an XACRO file for Publish/Subscribe to ROS-topics related to the velocity and the pose of a robot. For this study, the model of the Mecanum wheeled robot is established through the URDF file which is shown in Fig. 2. A camera link (the box placed on the top front of the robot body) and a lidar sensor link (the cylinder on the top back of the robot body) are installed on the robot base model [15].

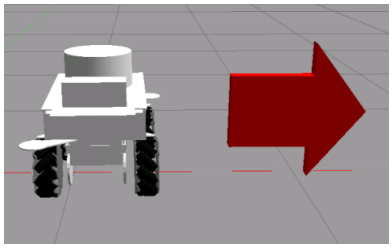

Fig 2. The Mecanum wheeled robot model (left) and the target red arrow model (right) in Gazebo.

Another format to represent models in ROS is Simulation Description Format (SDF). It provides an exhaustive account of everything, from the world level to the robot level and everything in between. It is simple to add new pieces and alter existing ones, therefore it has a high degree of scalability. SDFs are often used in the Gazebo simulation for non-ROS objects like walls, people, and items. The target red arrow in our project is set by an SDF file which is illustrated in Fig. 2.

C. Multi-view Following Setting

Without loss of generality, we study the two following modes in the simulated setting as shown in Fig. 3. The first mode is back-view following where the robot keeps up with the target arrow by moving ahead and maintaining a certain distance between the target and the robot. The second mode is side-view following, which refers to the robot following the target arrow from the subject's side by moving horizontally and keeping a certain distance from the target. In this mode, the robot's position is always on the right side of the target, and the depth camera is constantly facing the target where the orientation of the robot is always perpendicular to the target.

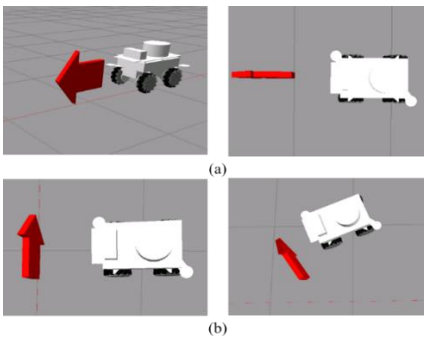

Fig. 3. Different following modes represented in 3D Gazebo simulation. (a) The back-view following mode where the robot's travel trajectory consistently aligns with the target arrow; (b) The side-view following mode where the Mecanum-wheeled robot achieves lateral movement and remains aligned with the target arrow.

D. Tracking and Controlling Algorithms

In the virtual environment, additive Gaussian noise will be introduced to the robot's observation data. The onboard Kalman filter will be employed to predict and track the target, consequentially improving the robustness, and maneuverability of the robot's tracking. Finally, the PID control algorithm will be utilized to enhance the robot's speed response and minimize the positional error. The KF involves a linear dynamical model for state prediction and a Gaussian noise model for system modeling. It updates the state value using the observed value of the current state and the estimated value of the state at the previous moment based on a recursive formula [16]. In the *KF*, the observed data are predicted at each time step k from the immediately preceding instant, by the state-transition model *F*, contaminated with process noise w_k, described by:

$$X_k = FX_{k-1} + w_k, w_k \sim N(0, Q_k), \quad (1)$$

where state $X = [x_k, y_k, \theta_k]^T$ includes 2D position (x, y) and yaw angle θ, and the observation state is:

$$y_k = HX_{k-1} + u_k, u_k \sim N(0, Q_Y), \quad (2)$$

where H is the observation model, u_k is the observation noise with zero mean and covariance Q_Y.

A PID controller is used for the speed control of the wheeled mobile robot thereby increasing stability of movement [17]. The PID controller computes the error signal which is the difference $e(t)$ between the measured output and the desired set point. The PID controller general equation is:

$$u(t) = K_P e(t) + K_i \int_0^t e(t) + K_d \frac{de(t)}{dt}, \quad (3)$$

where K_p, K_i, and K_d, denote the coefficients for the proportional, integral, and derivative terms, respectively.

IV. Simulation Setup

A. Data Generation

In the virtual environment, the corresponding robot tracking path is obtained by generating a target path. The target's path is determined by subscribing to the topic of the robot control system that governs the arrow's position and speed, where the ROS Topic /target/odom and /target/cmd_vel involves the target's orientation and velocity. After sending commands to the system regarding the target pose, the target robot can move to the desired location at a constant speed.

To simulate the observation error, a pre-generated array containing Gaussian noise values is sequentially added to the target trajectory published by /turtle_tf_broadcaster_node. After receiving the broadcast's location information, the listener sends movement commands to the Mecanum wheeled robot, thus the /turtle_tf_listener_node is picked as the channel for carrying the tracking and following algorithms. During robot tracking, the pose of both the target and the robot is published and recorded by the /odom topic. The whole process of data generation in ROS-Gazebo is depicted in Fig. 4.

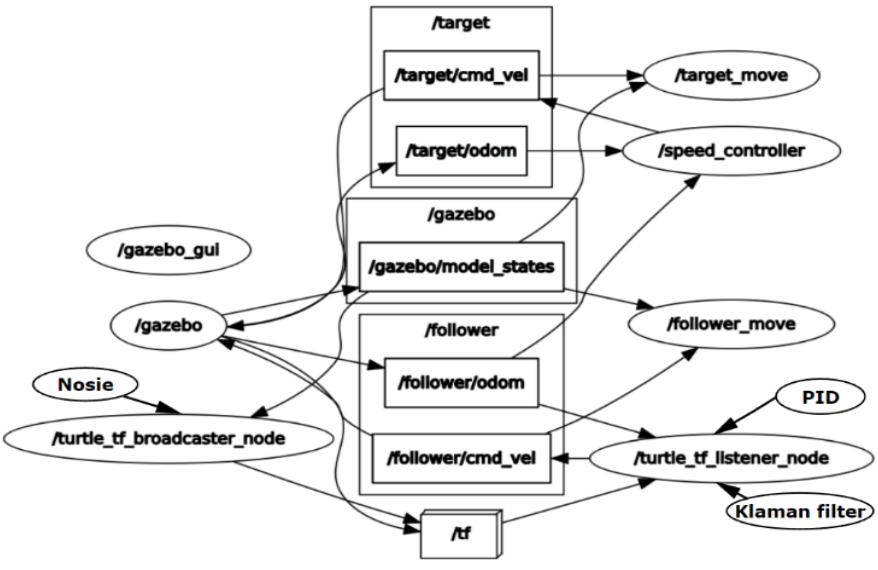

Fig. 4. Overview of data generation process in ROS Gazebo.

Through the above process, we captured the raw data of robot trajectories from back-view following and side-view following as well as those of the target. The trajectories generated from ROS are plotted in Fig. 5 (a) and (b).

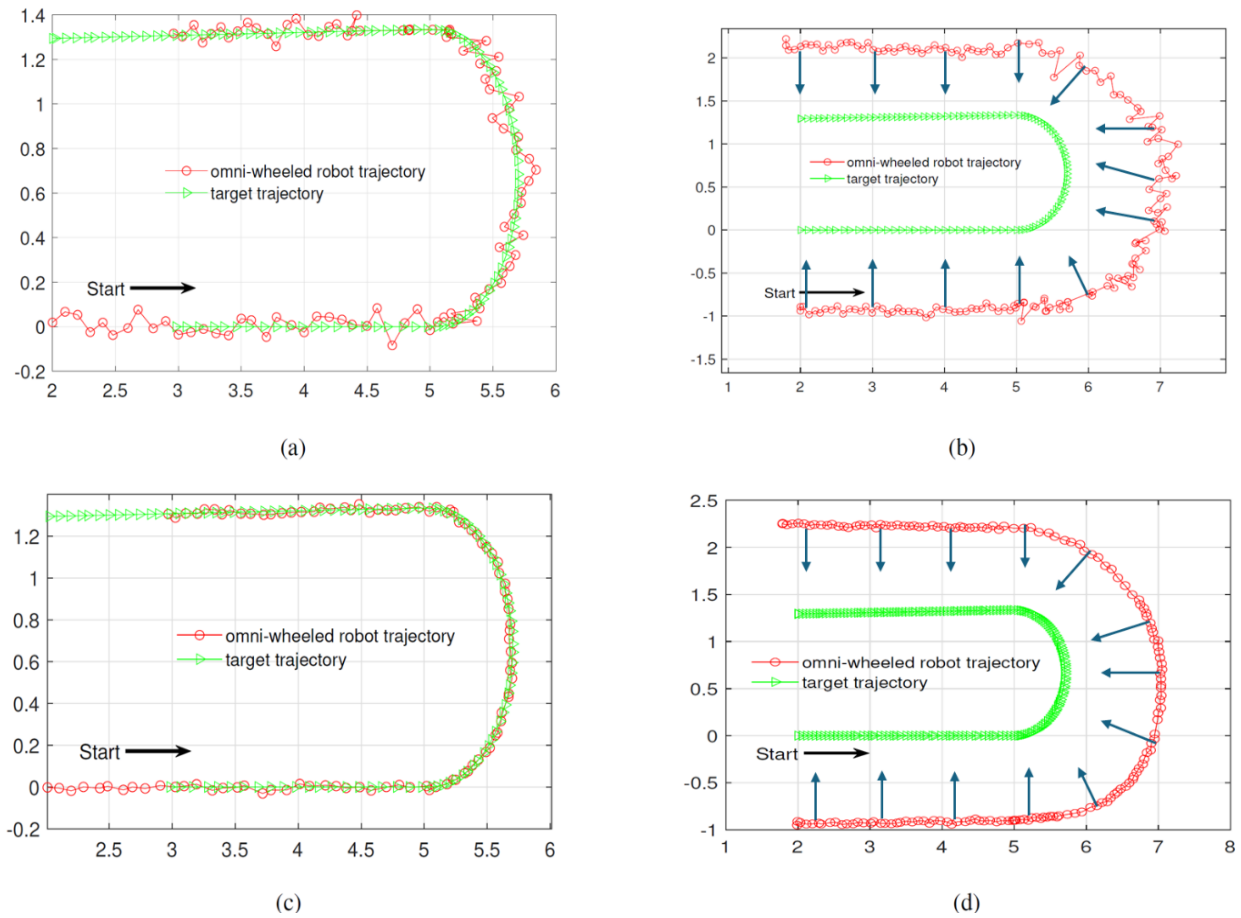

Fig. 5. Overall trajectories of the robot follower. (a) Back following without optimization; (b)Side-view following without optimization; (c) Back following with optimization; (d) Side following with optimization. With the arrow signifying the initial direction of the target trajectory and the blue arrows representing the front facing direction of the Mecanum wheeled robot.

B. *Tracking and Control Parameters*

In the virtual environment, the target moves forward with a linear velocity of 0.2 m/s at a constant speed and turns at an angular velocity of 0.1 rad/s. During the tracking process, the robot-to-target distance is constantly fixed at 1 meter, and the angle difference is kept at 0 in the back-view tracking mode and at $\frac{\pi}{2}$ in the side-view tracking mode.

During the simulation, the speed variables of the Mecanum wheeled robot include linear speeds along x and y axes and the angular speed. The PID parameters for each following mode are listed in Table I.

TABLE I. PID PARAMETERS IN THE SIMULATION

Following mode	Parameters	K_p	K_i	K_d
Back-view	Linear.x	2	0.01	0.2
	Linear.y	0	0	0
	Angular.z	1	0	0
Side-view	Linear.x	2	0.01	0.2
	Linear.y	1.8	0	0.1
	Angular.z	1	0	0

In addition to PID, two parameters of the Kalman filter are worth mentioning. The observation model is $H = [1,1,0]$ and the state-transition model F is expressed as:

$$F = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

where Δx and Δy denote the displacement per unit time in the x and y axes respectively.

C. *Quantitative Evaluation*

In the ROS environment, the robot's state primarily comprises the robot's orientation and position on the coordinate axis. Through different topics and nodes in the simulator, we can get the state of the following robot and the pose of the target to calculate the distance and angle difference between them. For instance, the state of the target is $(x_n^t, y_n^t, \theta_n^t)$ and the state of the following Mecanum wheeled robot is $(x_n^r, y_n^r, \theta_n^r)$ at nth frame in the simulation. θ_n^r is defined as the angle between the moving direction of the robot and the x-axis while θ_n^r is defined in the same way for the target. As shown in Fig. 6, we define the positional distance and angular difference in the following robot.

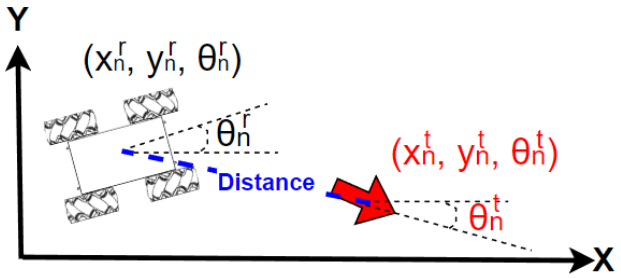

Fig. 6. 2D poses of a Mecanum wheeled robot and a target.

The distance between the following robot and the target at time n can be computed as

$$D_n = \sqrt{(x_n^r - x_n^t)^2 + (y_n^r - y_n^t)^2}. \quad (5)$$

The angular difference of the target and robot angles is defined as:

$$\Theta_n = \sqrt{(\theta_n^r - \theta_n^t)^2}, \quad (6)$$

where Θ_n is the difference of the angle between the robot and the target at time n. We aim to maintain the positional and angular differences between the following robot and the target close to their preset values in our simulator. Therefore, two metrics are used for performance evaluation. The first metric is the L1-norm error of positional deviation between the actual and preset distances through all frames which is defined as:

$$D_{err} = \frac{1}{N}\sum_{n=1}^{N}|D_n - D_0|, \quad (7)$$

where N is the total frame number, and D_0 is the expected target-robot distance. The second metric is the L1-norm error of angle deviation between the actual angle and the expected one for all time frames:

$$\Theta_{err} = \frac{1}{N}\sum_{n=1}^{N}|\Theta_n - \Theta_0|, \quad (8)$$

where the preset angle is θ_0. The configuration of tracking settings differs somewhat since we utilize multi-view tracking modes at various angles. While the distance D_0 is 1 m and the angle difference $\theta_0 = 0$ degrees during back-view following, these values are set to 1m and 90° during side-view tracking.

The entire tracking process is depicted in Fig. 7. The complete tracking circuit consists of a U-shaped track, where the initial point of the target is $(x_1^t, y_1^t, \theta_1^t)$ and is located at one extremity of the U-shaped track, while the robot is located at $(x_1^r, y_1^r, \theta_1^r)$. In the back-view following mode, the robot initially follows a linear trajectory, upon traversing a specific distance, the object will go onto a curved trajectory where the positions of the target are expressed by $(x_n^t, y_n^t, \theta_n^t)$ and $(x_m^t, y_m^t, \theta_m^t)$, and the poses of the robot are represented by $(x_n^r, y_n^r, \theta_n^r)$ and $(x_m^t, y_m^t, \theta_m^t)$, where $m > n$ and $m, n \in [1, N]$.

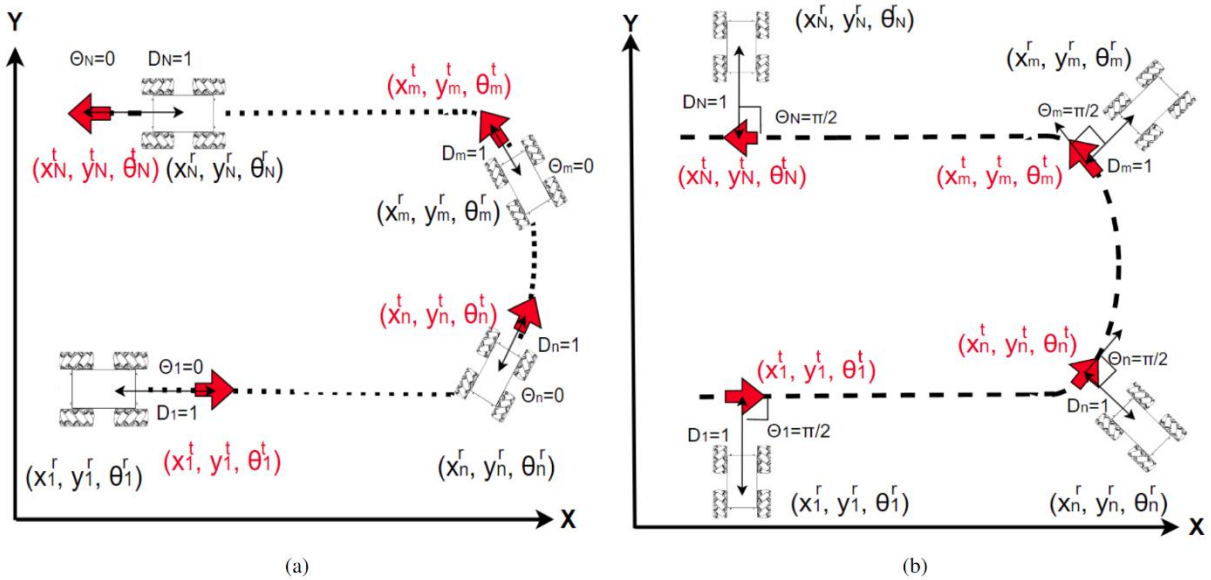

Fig. 7. 2D top-view of a robot tracking a target arrow in (a) back-view following and (b) side-view following modes along a U-shaped trajectory.

After exiting the curved section, the object will transition onto another linear path, ultimately coming to a halt at the opposite extremity of the U-shaped track, $(x_1^r, y_1^r, \theta_1^r)$, where N is the frame number of the whole following process. During the whole following process, the robot moves with its orientation always aligned with the direction of the red arrow. In addition to the back-view following mode, the side-view following mode involves the robot maintaining a steady position on the side of the red target arrow while tracking the running track of the target. The lateral movement capability of the robot is attributed to the unique properties of the Mecanum wheels. In the process of following, it is imperative to maintain a consistent moving angle between the robot and the target. Specifically, for a back-view perspective, the difference angle between the moving target and the robot θ_n should

be set to 0, while for a side-view perspective, it should be set to $\frac{\pi}{2}$. Additionally, it is essential to regulate the distance between the robot and the target, ensuring that it remains constant throughout the tracking process.

The overall curve of the robot is smoother after adjusting the Kalman filter and PID controller (Fig. 8(a) and (c) blue line), and the error curve created may also demonstrate that the overall distance error and angle error are minimized (Fig. 8(b) and (d) blue line). Fig. 8(a) illustrates an initial increase in distance error, attributed to the robot initiating tracking upon detecting target movement. Nevertheless, the implementation of PID and filtering algorithms enhances the robot's movement stability. Following optimization (Fig. 8(b) blue line), where the robot was initially trailing the target, the angle remains stable. During side-view tracking (Fig. 8(c) and Fig. 8(d)), it is evident that the error rises during the initial stages and turning process, subsequently decreasing upon returning to a straight route with the optimization.

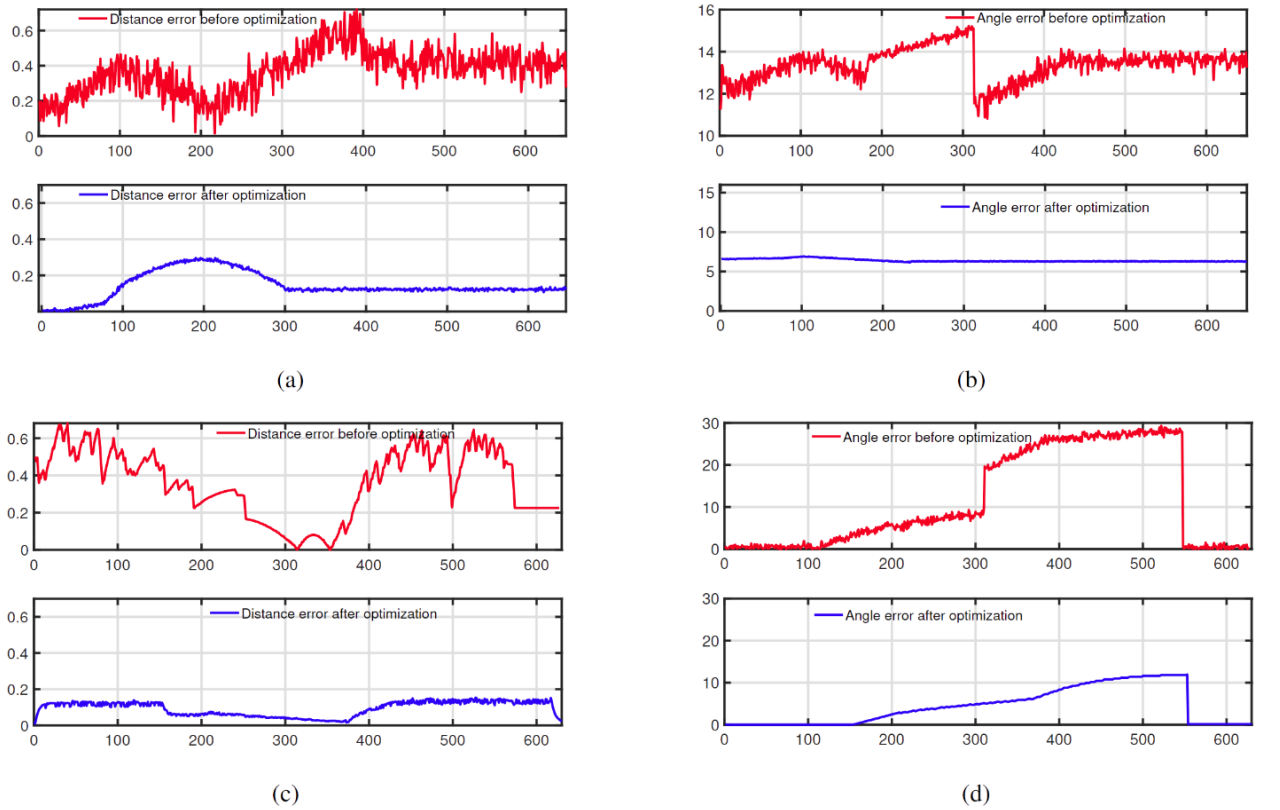

Fig. 8: Overall distance and angle errors of the robot follower before and after optimization. (a) back-view distance errors; (b) back-view angle errors; (c) side-view distance errors; (d) side-view angle errors.

Table II shows the numerical result of tracking errors before and after optimization. In terms of numerical results alone, the results of optimization for the back-view tracking are better than those of side-view tracking. However, given the more complex model control of side-view tracking and the percentage improvement, the accuracy of both tracking modes has improved by nearly 55% which indicates that our approach is crucial in optimizing side-view tracking as well. The results allow us to validate the virtual environment we created and demonstrate the benefits of a Mecanum wheeled robot in multi-view tracking based on the significant effectiveness after optimization.

TABLE II. TRACKING ERRORS FOR BACK-VIEW/SIDE-VIEW TRACKING

Tracking errors	W/o optimization	With optimization
D_{err} (back-view) (m)	0.22	0.12
Θ_{err} (back-view) $(\circ)$	12.2	5.3
D_{err} (side-view) (m)	0.32	0.14
Θ_{err} (side-view) $(\circ)$	13.4	6.2

V. CONCLUSION AND FUTURE WORK

In this study, we have developed a new ROS-based virtual environment with a Gazebo simulator specifically designed for the development and evaluation of multi-view tracking and control on an Mecanum wheeled robot. Unlike existing ROS-based simulations that primarily utilize differential drive systems, our simulator incorporates the crucial aspect of an omnidrive system. The inclusion of omni-wheels in multi-view human tracking robots significantly simplifies the vision and control algorithms, for example, obviating the need for a controllable gimbal. To showcase the effectiveness of our environment, we have implemented a PID controller and a Kalman filter for two-view target tracking and following. This work provides a much-needed research and experimental tool for the development and evaluation of a mobile Mocap system which can be used for human-centric multi-view gait analysis in a free-walking environment.

Our future research will be two-fold. First, we will extend the virtual ROS platform with the Gazebo simulator to support more realistic motion trajectories and more view-specific following modes along with improved filtering and tracking algorithms. We will release the source code of this ROS environment in the future to promote related research activities. Secondly, we will translate the validated filtering and tracking algorithms to a real-world prototype Mecanum-wheeled robot system for view-specific human following and quantitative gait analysis which support the mobile Mocap technology for various clinical gait analysis applications.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

ZL and GF developed the overall research plan, ZL conducted the research and prepared the initial manuscript. CE and NL analyzed the data and revised the manuscript under GF's supervision. All authors had approved the final version.

References

[1] S. Li, K. Milligan, P. Blythe, Y. Zhang, S. Edwards, N. Palmarini, L. Corner, Y. Ji, F. Zhang, and A. Namdeo, "Exploring the role of human-following robots in supporting the mobility and wellbeing of older people," *Scientific Reports*, vol. 13, no. 1, 6512, 2023.

[2] Z. Chen, H. Zhang, A. Zaferiou, D. Zanotto, and Y. Guo, "Mobile robot assisted gait monitoring and dynamic margin of stability estimation," *IEEE Transactions on Medical Robotics and Bionics*, vol. 4, no. 2, pp. 460–471, 2022.

[3] D. Guffanti, A. Brunete, and M. Hernando, "Development and validation of a ROS-based mobile robotic platform for human gait analysis applications," *Robotics and Autonomous Systems*, vol. 145, 103869, 2021.

[4] D. Guffanti, A. Brunete, M. Hernando, J. Rueda, and E. Navarro, "ROBOGait: A mobile robotic platform for human gait analysis in clinical environments," *Sensors*, vol. 21, no. 20, 6786, 2021.

[5] H. Taheri and C. X. Zhao, "Omnidirectional mobile robots, mechanisms and navigation approaches," *Mechanism and Machine Theory*, vol. 153, 103958, 2020.

[6] H. Zhang, Z. Chen, D. Zanotto, and Y. Guo, "Robot-assisted and wearable sensor-mediated autonomous gait analysis," in *Proc. 2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 6795–6802.

[7] R. Algabri and M.-T. Choi, "Deep-learning-based indoor human following of mobile robot using color feature," *Sensors*, vol. 20, no. 9, 2699, 2020.

[8] M. Gupta, S. Kumar, L. Behera, and V. K. Subramanian, "A novel vision-based tracking algorithm for a human-following mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 7, pp. 1415–1427, 2017. https://doi.org/10.1109/TSMC.2016.2616343

[9] J. Chen and W.-J. Kim, "A human-following mobile robot providing natural and universal interfaces for control with wireless electronic devices," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 5, pp. 2377–2385, 2019.

[10] J. Yuan, S. Zhang, Q. Sun, G. Liu, and J. Cai, "Laser-based intersection-aware human following with a mobile robot in indoor environments," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 354–369, 2018.

[11] R. K. Megalingam, R. Anandu, D. H. TejaAnirudhBabu, G. Sriram, and V. S. YashwanthAvvari, "Implementation of a person following robot in ROS-gazebo platform," in *Proc. 2022 International Conference for Advancement in Technology (ICONAT)*, IEEE, 2022, pp. 1–5.

[12] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, "Simulation environment for mobile robots testing using ROS and Gazebo," in *Proc. 2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, 2016, pp. 96–101. https://doi.org/10.1109/ICSTCC.2016.7790647

[13] A. Testa, A. Camisa, and G. Notarstefano, "ChoiRbot: A ROS 2 toolbox for cooperative robotics," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2714–2720, 2021.

[14] M. Ashmore and N. Barnes, "Omni-drive robot motion on curved paths: The fastest path between two points is not a straight-line," in *Proc. AI 2002: Advances in Artificial Intelligence: 15th Australian Joint Conference on Artificial Intelligence Canberra*, Australia, December 2–6, Springer, 2002, vol. 15, pp. 225–236.

[15] The ROS Construct. (2019). Using Gazebo Plugins to Simulate and Control Mecanum Wheels Robot. [Online]. Available: https://www.theconstruct.ai/use-gazebo-plugins-simulate-control-robot/

[16] A. Malaguti, M. Carraro, M. Guidolin, L. Tagliapietra, E. Menegatti, and S. Ghidoni, "Real-time tracking-by-detection of human motion in RGB-D camera networks," in *Proc. EEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 3198–3204.

[17] S. Shahin, R. Sadeghian, P. Sedigh, and M. T. Masouleh, "Simulation, control and construction of a four mecanum-wheeled robot," in *Proc. 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, 2017, pp. 0315–0319. https://doi.org/10.1109/KBEI.2017.8324993