Hand-Action Navigation & Detection Optimized Follower Framework (HANDOFF)

1st Jordan N. Brown
University of North Dakota
Grand Forks, ND, United States
brownjordan317@gmail.com

***Abstract*—This project presents the design and implementation of a low-cost, human-following robot capable of interpreting hand gestures to control its behavior. The system integrates multiple sensing modalities, including a forward-facing camera for person detection, a 2D LiDAR for distance estimation and obstacle avoidance, and vision-based human pose tracking for gesture recognition. YOLOv12 is employed for real-time person detection, while Google MediaPipe facilitates hand and body gesture recognition, allowing intuitive and responsive robot control. Sensor fusion of RGB and LiDAR data ensures accurate tracking even under partial occlusion or lateral movement. Gesture commands are mapped to key robot actions such as driving, target switching, and state control, enabling safe and interactive operation. The platform leverages affordable hardware, including a Raspberry Pi 5, a Pi Camera, and an RPLiDAR sensor, providing a cost-effective alternative to commercial systems. Experimental results demonstrate reliable target tracking, effective gesture-based control, and robust operation in dynamic environments, highlighting the system's potential for applications in assistive robotics, interactive robotic pets, and automated service platforms.**

***Index Terms*—gesture recognition, person tracking, robot control, LiDAR, embedded vision**

I. Introduction

The objective of this project is to develop a robot capable of locking onto an individual and autonomously following them while monitoring human gestures. These gestures can range from simple commands, such as instructing the robot to move closer or further away, to more complex actions, including returning to a home location or switching targets. While similar robotic systems exist commercially, they are often prohibitively expensive, with prices reaching thousands of dollars. This project aims to provide a cost-effective alternative that can achieve similar functionality using affordable hardware. Beyond the core capabilities demonstrated in this work, such a system has the potential to support a wide range of real-world applications. For example, assistive robots could follow elderly or mobility-impaired individuals, carrying items or responding to simple gesture-based commands. In logistics or warehouse environments, a follower robot could accompany workers, transporting tools or materials while enabling hands-free control. Additionally, the system's gesture recognition capabilities could enable mobile sign-language translation or support interactive educational platforms for teaching robotics, computer vision, and human–robot interaction. These examples highlight the versatility and practical utility of a low-cost, gesture-controlled following robot, motivating the development of an accessible platform capable of robust and intuitive human–robot interaction.

II. Platform Justification Sensor Specifications

A compact differential-drive base was selected for its mechanical simplicity and suitability for real-time following behavior. The RPLiDAR A1 provides full 360° scanning at a fraction of the cost of industrial units, and the Pi Camera 3 delivers adequate resolution for modern computer vision models without requiring specialized interfaces. Magnetic encoders were selected for their high resolution, durability, and compatibility with low-speed robotic actuation. Altogether, this configuration results in an inexpensive but highly capable research platform that supports multimodal perception, embedded machine learning, and real-time navigation. Hardware integration and mechanical assembly follow the UND_EDUBot build instructions [1].

TABLE I: Robot Hardware Specifications

Subsystem	**Component**	**Specification**
Compute	Raspberry Pi 5	8 GB RAM, Active Cooler
Lidar	RPLiDAR A1	2D Laser Scanner (360°)
Vision	Pi Camera Module 3	V2 (8 MP), CSI Interface
Actuation	ServoCity Motors	163 RPM Mini Econ Gear
Driver	Pololu Driver	TB67H420FTG Dual Driver
Odometry	Magnetic Encoders	AS5600 (12-bit Precision)
Power	3S UPS Module	18650 Li-Ion Array

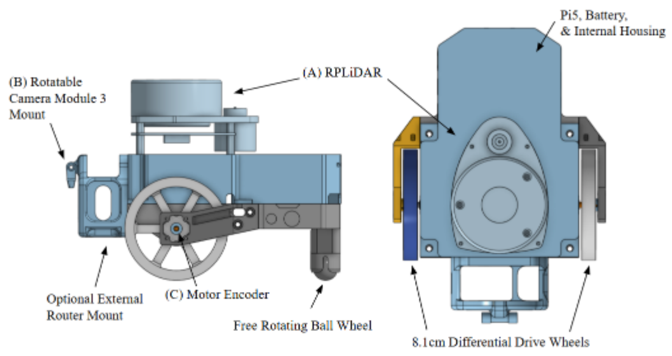

(B) Rotatable Camera Module 3 Mount
(A) RPLiDAR
Pi5, Battery, & Internal Housing
Optional External Router Mount
(C) Motor Encoder
Free Rotating Ball Wheel
8.1cm Differential Drive Wheels

Fig. 1: Annotated diagram of the robot showing key components.

A. RPLIDAR A1

The RPLiDAR A1 is a 360° 2D laser scanner used for obstacle detection and distance estimation. It provides continuous planar scans around the robot and is responsible for maintaining user distance and detecting environmental hazards. Key specifications include:

- **Scanning range:** 0.15 m to 12 m [2]
- **Scan coverage:** 360° omnidirectional
- **Scan rate:** up to 5.5 Hz
- **Interface:** 5 V supply via USB or external source; serial UART communication (3.3 V logic)

This LiDAR enables ground-level detection of legs and obstacles.

B. Raspberry Pi Camera Module 3

A Raspberry Pi Camera Module 3 serves as the primary visual sensor for person detection and gesture control. The camera feeds YOLOv12 for person detection and MediaPipe for gesture recognition. Key features include:

- **Image sensor:** Sony IMX708 HDR sensor [3]
- **Resolution:** up to 4608×2592 (approx. 11.9 MP)
- **Video modes:** 1080p@50 fps, 720p@100 fps, 480p@120 fps
- **Pixel size:** 1.4 µm
- **Field of view:** approximately $66° \times 41°$

The camera's HDR capability and auto focus allow for consistent gesture and person recognition under varying lighting and motion conditions.

C. AS5600 Magnetic Encoders

Wheel odometry is provided by AS5600 12-bit contactless magnetic encoders mounted perpendicular to each motor shaft. These sensors measure absolute angular position based on a diametric magnet, enabling precise velocity and rotation estimates. Specifications include:

- **Resolution:** 12-bit (4096 steps per revolution) [4]
- **Sensing method:** magnetic induction with no physical contact
- **Interfaces:** I2C digital output, or PWM/analog modes
- **Calibration:** programmable zero point and rotation range

III. System Limitations

While the Raspberry Pi 5 offers substantial improvements over previous Pi models, it still presents significant constraints when simultaneously running multiple perception pipelines. YOLOv12 person detection, MediaPipe gesture recognition, LiDAR processing, and encoder-based odometry together place substantial demands on CPU and memory resources. Under full load, frame rates decrease, thermal throttling may occur without active cooling, and memory pressure can limit the size and complexity of machine learning models.

Because of these limitations, much of the development and testing was conducted on a high-performance host workstation. The workstation used was a Dell Precision 3680 equipped with an NVIDIA RTX 4090 GPU, which provided sufficient computational headroom for rapid model training, high-resolution data processing, and experimentation with larger architectures that would be impractical on embedded hardware.

IV. Related Work

Human tracking has long been a focus of research in robotics, with early approaches relying on laser range-finding to estimate human positions [5]. Subsequent methods targeted resource-constrained embedded systems, optimizing tracking algorithms for low-power platforms [6]. At the same time, advances in machine learning enabled more robust and accurate human tracking [7]. Modern systems frequently employ sensor fusion, combining RGB imagery with LiDAR data to improve tracking performance, even on compact robotic platforms [8].

Vision-based person detection has been revolutionized by deep learning. Models such as YOLO perform real-time object detection with low computational overhead, while maintaining consistent object identities across video frames. YOLO's speed and accuracy make it particularly suitable for embedded robotics applications, where continuous tracking of a user is required without excessive resource consumption.

Gesture recognition enables intuitive human-robot interaction without specialized equipment. Earlier methods relied on silhouette-based estimation of joint positions [9], whereas modern machine learning approaches can infer full-body joint positions and motion from dense video streams [10]. Google's MediaPipe [11] framework extends these capabilities, providing real-time hand, body, and face tracking, as well as tools to build custom gesture recognition models and pipelines. MediaPipe allows developers to train domain-specific classifiers that can be integrated into a robotic system, enabling gestures to be translated into actionable commands in real time.

By combining YOLO-based person detection with MediaPipe gesture recognition, a robotic system can simultaneously track a user and interpret hand gestures, allowing for safe, responsive, and interactive control. This combination of sensor fusion, vision-based tracking, and gesture recognition forms the foundation for real-time, low-cost human-following robots capable of complex state control based on intuitive user input.

V. Methodology

The proposed system integrates multiple perception and control modules into a unified real-time following and gesture-interaction pipeline. As shown in Fig. 2, the robot receives two primary sensor inputs, RGB camera images and 2D LiDAR scans. The camera stream is processed by YOLOv12 for person detection and by MediaPipe for hand-gesture recognition, enabling the robot to identify the target user and interpret gesture-based commands.

Simultaneously, the LiDAR scans are processed using the DR-SPAAM algorithm to detect human legs and estimate the target's distance. The outputs of the vision and LiDAR pipelines are fused to produce a combined state estimate consisting of the target's angular offset and range. This fused estimate is fed into a PID-based follower controller that computes linear and angular velocity commands for the differential-drive robot.

Gesture-based commands operate as a parallel state-control channel, allowing the user to switch targets, pause following, or drive the robot manually using hand signs. This combination of multimodal sensing, fusion, and classical closed-loop control forms the basis of the system's interactive behavior.

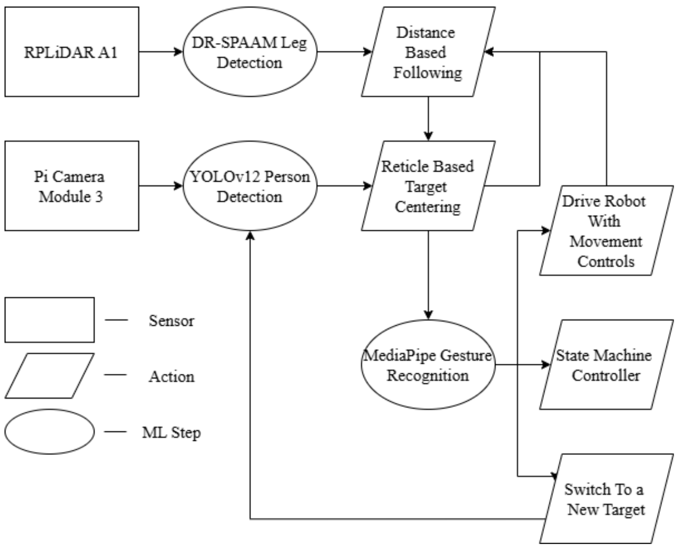


Fig. 2: Visual overview of the systems methodology and p.

A. Camera-Based Person Detection

Vision-based person detection for this project is handled using the pretrained YOLOv12 model provided by Ultralytics [12]. This version of YOLO was selected for its ability to perform real-time inference with relatively low computational overhead, making it well-suited for embedded or robotics applications. YOLOv12 also incorporates improved attention mechanisms that help maintain consistent object identities across frames, which is essential for reliable person tracking. Using YOLO's identification capabilities, a targeting pipeline was developed in which a user positions themselves within a reticle displayed on the camera feed. When a person remains inside this reticle for a specified duration (three seconds in this implementation), their ID is designated as the locked target. After lock-on, the user can move anywhere in the frame, and the system will continue tracking them while ignoring other individuals who appear. The locked user's displacement from the reticle center is then converted into angular motion commands, enabling the robot to smoothly adjust its orientation and keep the target centered in view.

To compute the commanded turn velocity, we normalize the signed pixel displacement and map it to a minimum magnitude for responsiveness while capping the maximum speed. Let d be the signed displacement in pixels (positive to the right) and $d_{\max}$ the maximum expected displacement. Define:

$$d_{\text{norm}} = \frac{|d|}{d_{\max}}, \quad d_{\text{norm}} \in [0,1]. \tag{1}$$

The commanded magnitude is:

$$v_{\text{mag}} = 0.5 + d_{\text{norm}}(1.5 - 0.5) = 0.5 + d_{\text{norm}}, \tag{2}$$

and the signed command is:

$$v = \text{sign}(d) \cdot \text{clip}\left(v_{\text{mag}},\, 0.5,\, 1.5\right), \tag{3}$$

where $\text{clip}(x, a, b) = \min(\max(x, a), b)$. This yields $v \in [-1.5, -0.5] \cup [0.5, 1.5]$ depending on the sign of d. Where d is the distance in pixels from the center of the reticle that a user is observed.

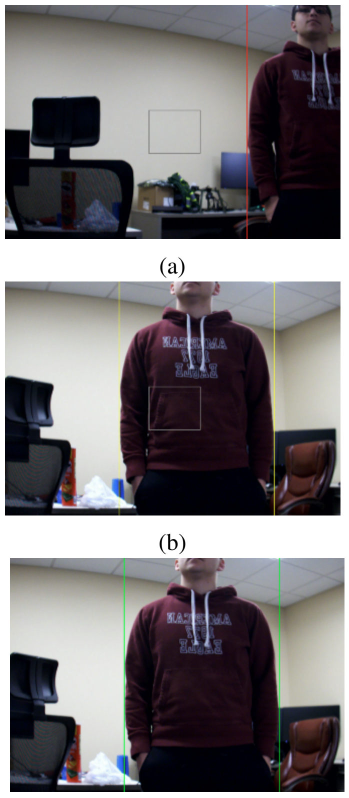
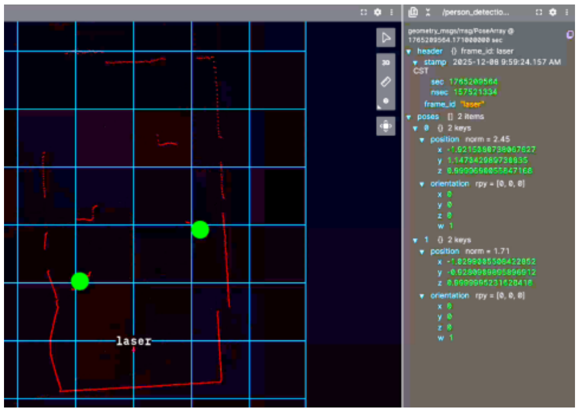

(a)
(b)
(c)

Fig. 3: (a) Person in camera view but not in reticle. (b) Person in camera view and within reticle. (c) Person is now the locked target.

B. 2D LiDAR Person Detection

Because the robot used in this project has a height under 20 cm, detecting human legs in 2D LiDAR scans provides a reliable method for estimating the user's distance relative to the system. For this purpose, the DR-SPAAM method [13] was selected due to its state-of-the-art performance in leg detection within LiDAR data. By pairing LiDAR measurements with the forward-facing camera's visual input, the system can robustly determine the distance to the target, enabling accurate tracking even when the user moves laterally or partially out of the camera's field of view.



```
geometry_msgs/msg/PoseArray @
1765209564.171000000 sec
▾ header  {} frame_id: laser
  ▾ stamp  2025-12-08 9:59:24.157 AM
  CST
      sec  1765209564
      nsec  157521334
    frame_id  "laser"
▾ poses  [] 2 items
  ▾ 0  {} 2 keys
    ▾ position  norm = 2.45
        x  -1.9215888738067627
        y  1.147842989738835
        z  0.9999698055847168
    ▾ orientation  rpy = [0, 0, 0]
        x  0
        y  0
        z  0
        w  1
  ▾ 1  {} 2 keys
    ▾ position  norm = 1.71
        x  -1.8299885508422852
        y  -0.9280098895896912
        z  0.9999995231628418
    ▾ orientation  rpy = [0, 0, 0]
        x  0
        y  0
        z  0
        w  1
```

Fig. 4: DR-SPAAM leg detections visualized with markers using the ROS 2 visualization program Foxglove.

C. *Merging LiDAR and RGB Person Detection*

The locked target centering system ensures the robot's orientation is continuously corrected, allowing the LiDAR scan to be restricted to the area directly in front of the robot with a defined tolerance. Detections within this zone are then processed to assign a precise distance to the target. By fusing LiDAR and RGB information in this way, the system combines the high spatial accuracy of LiDAR with the identification capabilities of vision-based detection, improving overall reliability in tracking a moving user.

D. *Robot Follower Using Sensor Fusion*

To maintain a stable distance and orientation relative to the user, the robot employs a feedback loop modeled after a proportional–integral–derivative (PID) controller. In a conventional PID framework, the controller receives a measured system state, compares it against a desired reference, computes the resulting error, and generates a corrective control signal. Our system follows this same structure: fused sensor measurements serve as the feedback signal, the desired following distance and orientation provide the reference, and the resulting positional error drives the PID computation.

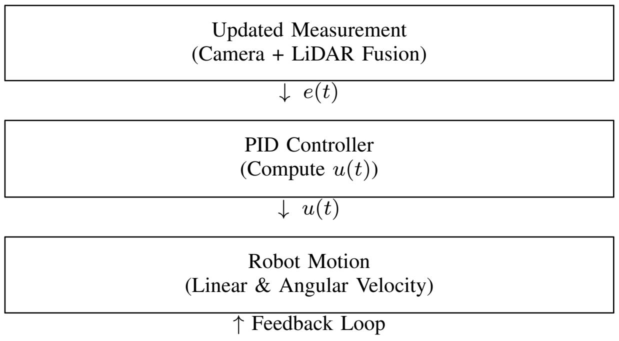


Fig. 5: PID-style control loop for robot following.

The general PID control law is defined as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t), \tag{4}$$

where $e(t)$ represents the displacement error between the robot and the target, K_p is the proportional gain, K_i is the integral gain, and K_d is the derivative gain. The output $u(t)$ is converted into the robot's linear and angular velocity commands.

The error term follows the standard PID definition but is computed from a multimodal fused measurement rather than a single sensor source. The angular component of the error comes from the camera's horizontal pixel offset (representing left–right misalignment), while the distance component is obtained from LiDAR range readings (representing forward–backward deviation). These complementary signals form the system's instantaneous state estimate, and the difference between this estimate and the desired following posture constitutes $e(t)$ in the PID equation.

By continuously evaluating the proportional, integral, and derivative contributions of this fused error, the controller stabilizes both distance and orientation in real time. As in typical PID applications, this allows the robot to smoothly adjust its motion even as the user accelerates, decelerates, or changes direction. This demonstrates a direct correspondence between our implementation and the canonical PID feedback control structure.

E. *Vision-Based Human Joint Tracking*

Human pose tracking is supported by the YOLO model used for detection, specifically the YOLO12-pose variant [12]. While this model effectively identifies full-body key points, it does not include detailed hand tracking or the ability to recognize specific hand gestures. To address these limitations, Google's MediaPipe Solutions [**?**] offers a more comprehensive alternative. MediaPipe provides full-body pose estimation comparable to YOLO but also includes hand-landmark tracking and optional face mapping. Most importantly, MediaPipe includes a dedicated framework for training custom hand-gesture recognition models, allowing developers to create classifiers tailored to their applications. Once trained, such a model can be seamlessly integrated into a pipeline that interprets gestures and issues corresponding to robot commands, enabling precise and reliable gesture-based control.

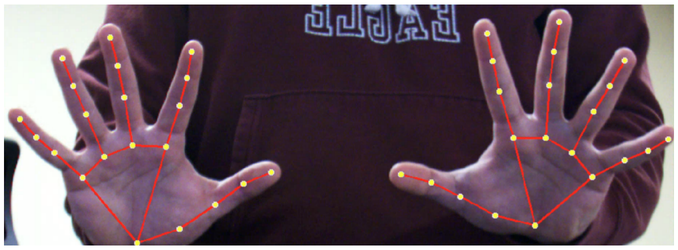


Fig. 6: Human hands with pose markers on the joints marked using Google's MediaPipe

VI. Training Hand Gesture Recognition

1) Training Method: Training in the gesture-recognition model can be performed entirely using the MediaPipe Model

Maker package [?]. This framework streamlines the end-to-end process by handling data ingestion, model construction, training, and export within a unified workflow. Users can adjust a wide range of hyperparameters, such as learning rate, batch size, number of epochs, and model architecture options. Users may also supply any number of custom images to tailor the model to their specific application. This combination of ease of use and configurability makes MediaPipe Model Maker well-suited for developing high-performance, domain-specific gesture-recognition models.

2) Dataset Selection: For hand-gesture recognition, the American Sign Language (ASL) alphabet was selected as the basis for the model's classes. While the full ASL alphabet contains 26 letters, J and Z require dynamic motions rather than static hand poses, so they were excluded, resulting in 24 static gesture classes for training. Three datasets [?], [?], [?] were used as the foundation. Each dataset includes labeled images of all 24 ASL letters captured under varied backgrounds, lighting conditions, and camera perspectives to support robust generalization. To further increase diversity and reduce overfitting, extensive data augmentation was applied, including random flips, rotations, color jittering, and homographic warping. These augmentation strategies were selected to emulate realistic operating conditions such as rapid lighting changes, hand pose variability, motion blur during walking, and sensor noise introduced by the Raspberry Pi camera. After combining the original images with all augmented variations, the final training set contained more than 680,000 labeled samples.

3) Training Results: After training, the gesture-recognition model demonstrates strong overall performance, with most letters achieving high true-positive rates and showing very little confusion with other gestures. At the same time, a substantial portion of samples are classified as "none," a trend clearly visible in the confusion matrix. While this behavior might appear concerning in a typical classification task, it is desirable for this application: because the gestures are used to issue robot control commands, it is far safer for the system to return "no gesture detected" than to misinterpret a hand sign and trigger an unintended robot action. This trade-off is further supported by the model's evaluation of metrics. With precision above 95%, the classifier almost never assigns the wrong label when it does recognize a gesture. However, the recall of just over 40% indicates that the system often chooses not to classify a gesture at all. In this context, high-precision, low-recall behavior aligns well with the safety-first requirements of gesture-based robot control.

A. Translating Gestures to Robot Controls

Once the gesture-recognition model has been trained, its outputs can be integrated into the robot operating system and code base. With 24 gesture classes per hand plus an additional "none" class, the system can represent up to 625 distinct two-hand combinations (or 25×25) when hand identity is considered, and 252 combinations if hand order is ignored. For this project, only a small subset of these combinations

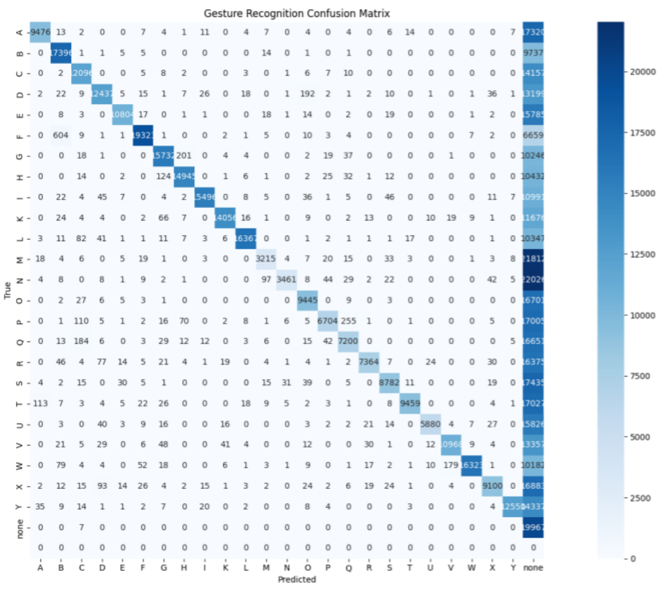


Fig. 7: Hand gesture training resultant confusion matrix.

has been implemented as proof of concept. The recognized gestures have been mapped to key robot functions, including driving commands, target switching, and overall state control.

1) Driving the Robot with Hand Gestures: Driving the robot using hand gestures proved to be one of the qualitatively reliable tasks for translating gestures into robot control. The robot subscribes to a ROS 2 topic for linear and angular velocity commands, which allows gesture inputs to be sent via a straightforward publisher program. For basic testing, only the V and "none" gestures were used. Since the robot is a differential-drive platform, each wheel can be controlled independently, resulting in the following control mappings:

TABLE II: Example Gesture Control Mapping

Gesture (L)	**Gesture (R)**	**Expected Control**
V	none	Left wheel rotates forward (turn right)
none	V	Right wheel rotates forward (turn left)
V	V	Both wheels rotate forward (drive forward)
none	none	Both wheels stop

During testing, this set of commands allowed smooth operation of the robot, comparable to using a standard ROS 2 teleop controller. The primary limit observed was latency when controlling the robot over a low-bandwidth network connection to a host computer.

2) Target Switching with Hand Gestures : A user within the robot's field of view (FOV) may wish to stop being followed. This can be accomplished by switching the robot into its unlocked state. Traditionally, this would require restarting the program or moving rapidly out of the robot's FOV. However, with gesture-based control, this action can be performed seamlessly and without abrupt movements. By holding up an "L" gesture with both hands, forming a goalpost shape, the robot exits its locked state. This allows another user to enter the frame and initiate the target-locking process.

3) State Control Machine with Hand Gestures: Beyond locking and unlocking the robot's target, gestures can be mapped to a wide range of state controls, enabling intuitive and versatile interaction. For example, a user could issue a gesture to pause or resume following, switch targets, or even trigger predefined behaviors such as performing a "trick" like those of a robotic pet. By defining a library of gestures and associating each with specific robot states or actions, the system provides a flexible interface that allows the robot to respond dynamically to user input without the need for physical control or a keyboard. This framework makes gesture-based control a powerful tool for both functional operation and interactive demonstrations.

VII. Discussion

The system developed in this project demonstrates that modern computer vision and LiDAR processing techniques can reliably perform person-following and gesture-based control. The fusion of YOLO-based person detection with 2D LiDAR distance estimation proved effective in maintaining both orientation and spacing even under partial occlusions or lateral movement. The results indicate that robust human–robot interaction does not require expensive or specialized sensors, but instead can be achieved through thoughtful algorithm design and multimodal perception.

However, several challenges were encountered. First, the Raspberry Pi 5, struggles to run gesture recognition, person detection, and LiDAR processing simultaneously at high frame rates. Real-time performance on embedded platforms remains sensitive to throttling, model size, and resource contention across CPU cores. Second, the lack of 3D depth sensing limits the robot's ability to infer user height, identify complex hand poses in low-light environments, or detect obstacles at head level. Finally, gesture recognition accuracy depends heavily on lighting conditions and camera orientation, and certain gestures introduce ambiguity when the user is partially outside the frame. These limitations provide valuable insight into the design considerations required for future iterations of interactive robotic systems.

VIII. Future Work

There are several meaningful extensions that would significantly improve the capability, reliability, and overall usability of the system. One immediate direction is offloading computationally expensive tasks—such as YOLO inference—to dedicated accelerators like the Google Coral TPU or NVIDIA Jetson platforms. This would enable higher frame rates, more responsive gesture detection, and improved real-time performance on embedded hardware.

Another promising improvement involves adopting 3D sensing modalities, such as stereo cameras or depth sensors. Incorporating depth information would enhance gesture recognition, improve obstacle avoidance, and allow the robot to better interpret complex human motions. A stereo vision system could also estimate user distance directly through disparity, effectively replacing the need for a LiDAR sensor in the follower PID loop. This would simplify the hardware architecture, reduce power consumption, and unify all perception tasks under a single sensing modality. Similarly, implementing a more advanced multi-person tracking system could enable dynamic user switching, group following, or crowd-aware navigation.

Future work may also expand the gesture vocabulary by training a custom MediaPipe gesture classifier using a domain-specific dataset. This would allow more expressive control, such as directional pointing, stop-and-go commands, or compound gestures involving both hands. Additional improvements include integrating full SLAM for autonomous mapping, enabling long-term navigation, and adding wireless communication so the robot can provide feedback to the user via audio, LEDs, or mobile notifications.

Another important extension is adapting the system for integration into specific commercial and practical use cases. For example, the robot could be specialized into an autonomous golf caddie capable of following a player across diverse terrain while managing equipment. Similarly, a retail-oriented variant could assist customers in stores by following them, providing product information, or carrying purchased items. In industrial logistics, the platform could be adapted into a mobile payload carrier, autonomously trailing a worker and transporting tools or materials. In assistive technology applications, gesture-responsive following behavior could support individuals with mobility impairments, enabling hands-free personal assistance in homes or care facilities. Tailoring the perception and control modules for these domain-specific applications would further validate the system's practicality and broaden its real-world impact.

IX. Conclusion

This project demonstrates the successful development of a low-cost, gesture-controlled human-following robot that leverages multimodal sensing for robust and intuitive interaction. By combining YOLOv12-based person detection, MediaPipe gesture tracking, and 2D LiDAR distance estimation within a PID-style control framework, the system achieves reliable tracking, navigation, and gesture-driven behavior control. The use of accessible hardware such as the Raspberry Pi 5 and RPLIDAR A1 highlights the feasibility of constructing capable interactive robots without the financial burden of commercial platforms.

Experimental testing shows that the robot can maintain lock-on to a target, interpret user gestures in real time, and modulate its motion based on fused sensor data. While computational limitations and sensor constraints introduce challenges, the system provides a strong foundation for future work in human–robot interaction, embedded machine learning, and autonomous mobile robotics. Overall, the results demonstrate that multimodal perception and gesture-based control can enable safe, responsive, and engaging robotic behavior using affordable components.

References

[1] Merila, J. (2025). UND_EDUBot: Educational Robot Build Instructions. GitHub Repository. https://github.com/JohnMerila/UND_EDUBot

[2] SLAMTEC. (2025). RPLIDAR A1 360° 2-D Laser Scanner — Specification. https://www.slamtec.com/en/lidar/a1

[3] Raspberry Pi Ltd. (2025). Raspberry Pi Camera Module 3 Sensor Assembly — Product Brief. https://datasheets.raspberrypi.com/camera/sensor-assembly-product-brief.pdf

[4] WWZMDiB. (2024). WWZMDiB 4Pcs AS5600 Magnetic Encoder 3.3V 12bit high Precision Magnetic Induction Angle Measurement Sensor Module, Mainly Used to Obtain Information Such as Progressive Motor Speed. Amazon.com: Industrial & Scientific.

[5] Google LLC. (2025). MediaPipe Solutions Hand gesture recognition model customization guide. Google AI Edge. https://ai.google.dev/edge/mediapipe/solutions/customization/gesture_recognizer

[6] Chung, W., Kim, H., Yoo, Y., Moon, C. B., & Park, J. (2011). The detection and following of human legs through inductive approaches for a mobile robot with a single laser range finder. IEEE Transactions on Industrial Electronics, 59, 3156–3166. https://doi.org/10.1109/TIE.2011.2170389

[7] Agarwal, A., & Triggs, B. (2005). Monocular Human Motion Capture with a Mixture of Regressors. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) – Workshops, 72. https://doi.org/10.1109/CVPR.2005.496

[8] Peng, Z., Xiong, Z., Zhao, Y., & Zhang, L. (2023). 3-D Objects Detection and Tracking Using Solid-State LiDAR and RGB Camera. IEEE Sensors Journal, 23(13), 14795–14808. https://doi.org/10.1109/JSEN.2023.3279500

[9] Abrego-González, J., Aguirre, E., & García-Silvente, M. (2024). People detection on 2D laser range finder data using deep learning and machine learning. Proceedings of the XXIV Workshop of Physical Agents, 235–249.

[10] Fod, A., Howard, A., & Mataric, M. A. J. (2002). A laser-based people tracker. Proceedings of the IEEE International Conference on Robotics and Automation, 3024–3029. https://doi.org/10.1109/ROBOT.2002.1013691

[11] Girdhar, R., Gkioxari, G., Torresani, L., Paluri, M., & Tran, D. (2018). Detect-and-Track: Efficient Pose Estimation in Videos. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 350–359. https://doi.org/10.1109/CVPR.2018.00044

[12] Ultralytics Inc. (2025). Models Supported by Ultralytics [Documentation]. https://docs.ultralytics.com/models/

[13] Jia, D., Hermans, A., & Leibe, B. (2020). DR-SPAAM: A Spatial-Attention and Auto-regressive Model for Person Detection in 2D Range Data. arXiv:2004.14079. https://arxiv.org/abs/2004.14079