A Vision-Driven End-User Robot Programming Method Based on the Alignment between Body Language and Motion Commands

Shipeng Lyu, Shenzeng Huo, Wanyu Ma, Guodong Guo, *Senior Member, IEEE* and David Navarro-Alarcon, *Senior Member, IEEE*

***Abstract*—This work is motivated by the complexity of conventional robot programming methods, which require a deep understanding of both robotics and software development. This situation presents significant challenges for non-technical end-users who need to program a new robotic task. To address this issue, we propose a new robot programming method that enables humans to specify robot motion tasks using simple and intuitive body-language commands. To this end, we propose a triadic task decomposition framework to decompose complex robot motion tasks into three basic task units, i.e., location, operation, and motion. These units are paired with a set of body-language cues that enable end-users to specify a variety of actions and motions, providing an intuitive alternative to traditional robot programming methods. To program these units, we develop a multimodal fusion-based instruction-parsing algorithm with a dual-phase visual architecture, enabling temporally aware body-language recognition and high-accuracy cross-modal object localization. Furthermore, a semantics-adaptive task execution system is proposed which incorporates an intent reasoning module that dynamically handles end-users' non-standard programming sequences and incomplete task logic, enhancing programming success rates and system usability. We evaluate the proposed method on a dual-arm robotic platform and a humanoid robot through various pick-and-place tasks, and compare its performance with that of several baselines from the literature.**

***Index Terms*—Human-robot interaction; End-user programming; Robot manipulation; Gaze; Hand gestures.**

I. Introduction

ROBOT programming skills have recently become increasingly important due to the growing use of robots in our daily lives. Moreover, these automatic machines are no longer confined to traditional manufacturing environments. Nowadays, robots are used to perform a wide range of tasks, e.g., delivering tools [1], cleaning [2], food catering [3], and shopping mall assistance [4]. Despite their widespread use across different environments, robots are still typically programmed and configured by technically trained personnel (see Fig. 1). However, interactions with these systems by non-technical end-users are typically reduced to a small subset of predefined actions that do not allow for specifying new motion tasks. To alleviate this problem, coding-free approaches have been introduced to enable non-technical end-users to specify tasks in a direct and intuitive manner, thereby broadening access to robotic systems.

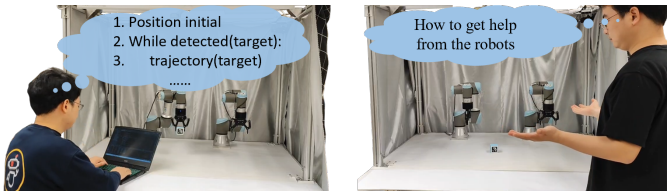

Fig. 1. (a) Robotics engineers typically operate robots and specify tasks through programming APIs. (b) Non-technical end-users encounter significant challenges in interacting and programming with robots.

Many coding-free programming methods have been proposed for different robotic scenarios [5], particularly in manufacturing [6], [7], where they enable human operators to program tasks using speech and text [8], body language [9], [10], and expert demonstrations [11]. Although these approaches provide intuitive channels to command robots, there are still several limitations that restrict their applicability in daily tasks. For example, these methods generally require the use of specialized hardware, e.g., VR glasses [12], [13], or wearable devices [14]–[16], which may not be practical in many real-world situations. Other methods, such as speech-based interfaces built on large language models (LLMs) [17], still require intermediate to advanced technical skills (e.g., prompt engineering and fine-tuning [18]) to specify new tasks and achieve reliable performance in personalized settings. A key challenge of vision-based solutions lies in interpreting robot commands from visual observations of human body gestures [19], [20]. Many of these approaches neglect the semantic alignment between the meaning conveyed by gestures and the intended task instruction, which often leads to user confusion and limits their practical performance.

To address these challenges, we propose a novel vision-driven method that decomposes complex interaction tasks into a set of subtasks, as illustrated in Fig. 2. This decomposition not only simplifies the interaction process but also enables more precise control over individual actions. By leveraging robot vision, our method eliminates the need for humans to wear or operate additional devices, thereby significantly reducing the complexity of human-robot interaction and en-

This work is supported in part by the Research Grants Council (RGC) of Hong Kong under grant C4042-23GF, and in part by the PolyU-EIT Collaborative PhD Training Programme under application number 220766263. *Corresponding author: David Navarro-Alarcon.*

S. Lyu, S. Huo, W. Ma, and D. Navarro-Alarcon are with the Department of Mechanical Engineering, The Hong Kong Polytechnic University (PolyU), Kowloon, Hong Kong. (e-mail: shipeng.lyu, wanyu.ma, kyle-sz.huo@connect.polyu.hk, dnavar@polyu.edu.hk)

S. Lyu, and G. Guo are with the Ningbo Institute of Digital Twin, Eastern Institute of Technology (EIT), China. (e-mail: gdguo@eitech.edu.cn)

hancing usability. End-users can provide instructions through simple eye gazes and hand gestures, which offer a natural and intuitive modality for communication. To ensure clarity and effectiveness, we designed a set of predefined body-language commands based on common human expressions, so that each gesture naturally conveys its intended robotic action. This design choice not only aligns with human intuition but also creates a user-friendly programming experience, making the system accessible to users with minimal technical expertise. Compared with existing works, we summarize the original contributions of the paper as follows:

1) We propose a triadic task decomposition framework that simplifies complex task programming through intuitive workflows for non-technical end-users, while enhancing programming efficiency via dynamic composition of semantically aligned instructions.
2) We develop a multimodal fusion-based instruction-parsing algorithm employing a dual-phase visual architecture, achieving temporally aware gesture recognition and high-accuracy cross-modal object localization.
3) We create a semantics-adaptive task-execution system incorporating an intent reasoning module that dynamically handles end-users' non-standard programming sequences and incomplete task logic, thereby improving programming success rates and system usability.

II. Related Work

Gaze and gestures have emerged as the predominant modalities employed by humans in contemporary research on human-robot interaction (HRI). These studies explore the use of one or both modalities to achieve specific objectives in robot programming by leveraging the distinct advantages of each.

As gestural communication is a fundamental modality of human interaction, it is natural to incorporate gestures into HRI as a robot-programming paradigm. Existing implementations cover several main application scenarios, including spatial target designation [19], trajectory specification [20], and elementary object manipulation tasks [21]. Although these approaches have achieved notable success in constrained environments, important limitations remain in terms of semantic alignment. A semantic mismatch often arises between the natural meaning of gestures in human communication and the predefined meanings assigned to them in robotic systems. This mismatch is particularly evident in arbitrarily designed gesture vocabularies, which may conflict with end-users' intuitive expectations and lead to interaction patterns that require explicit training rather than relying on natural communication habits.

Gaze interaction has become an important approach in human-robot programming systems, particularly for object localization in unstructured environments [22]. Current gaze-based systems support tasks such as target localization [23]–[25], attention-driven manipulation sequencing [26]–[28], and intent-aware navigation planning [29]. Despite their success in task execution, existing gaze-based systems still face inherent limitations in precise 3D position estimation when relying solely on gaze information. Recent multimodal approaches attempt to mitigate this through hybrid gaze-gesture fusion [30], demonstrating 4.48% improvement in spatial inference accuracy compared to unimodal baselines. However, such solutions often require complex multimodal sensor-fusion frameworks, multi-view camera calibration, and highly controlled laboratory environments. These requirements impose significant constraints on real-world deployment.

Recent advances in vision-based human activity recognition have further expanded the potential of intuitive robot programming. For example, a variety of methods have been proposed for gesture recognition [31]–[34] and gaze estimation [35]–[38]. These works highlight the importance of aligning semantic intent with sensory inputs, which is also a central principle of our approach. However, many existing methods focus on isolated modalities or rely on controlled environments. In contrast, our method leverages multimodal fusion and onboard robot vision to improve robustness. Additionally, unlike template-based gesture recognition [33], our spatiotemporal model captures dynamic gesture transitions, reducing misclassification during continuous programming.

III. Methodology

We propose a vision-driven robot programming method for non-technical end-users that aligns body-language commands with robot motion commands. As shown in Fig. 2, the end-user decomposes the interaction task according to their intention and understanding of the task. Then, the end-user can program the robot based on predefined body-language commands. During programming, the robot continuously detects the end-user's actions through its onboard vision system. Once the end-user finishes programming, the robot reasons over the detected motion commands and generates actions to complete the task.

A. Interaction Task Definition

To enable end-users to intuitively program robotic tasks, we propose a triadic task model in Eq. 1 that decomposes complex interactions into fundamental components: Location (L), Operation (O), and Motion (M). This model aligns with the natural human cognitive process of task decomposition, where individuals typically first identify where to act (location L), then what to do (operation O), and finally how to move (motion M). For example, in a "Pick-and-Place" task, the robot first moves to the object's initial location and executes a "pick" operation to grasp the object. It then follows a collision-free path to the target location, where it releases the object through a "place" operation.

$$I_i \in I = \{L, O, M\}. \tag{1}$$

By decomposing the HRI task into these components, end-users can systematically program each step through predefined body-language commands without coding low-level robotic actions. This approach mimics how humans teach others by breaking tasks down into sequential, contextually simple subtasks. This makes the approach accessible to non-technical end-users. Notably, not all tasks require all three elements. A task may involve only motion (e.g., "move from point A to B") or only operation (e.g., "rotate an object in place"). The flexibility of this model allows users to combine components as needed, thereby ensuring adaptability across diverse scenarios.

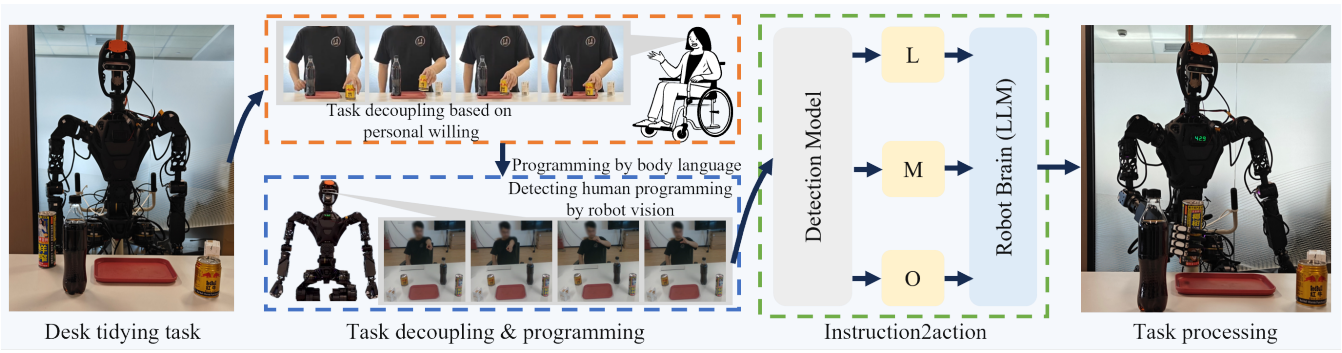

Fig. 2. Architecture of vision-driven robot programming method for non-technical end-users. End-users decompose the task based on their intentions and understanding, and program each decomposed action using predefined body-language commands. After detecting human instructions, the robot uses pretrained models to translate them into actions.

B. Programming Commands Design

To translate human intentions into robot actions, we design a set of intuitive body-language commands aligned with natural human communication. These commands are categorized into task commands (Pick, Place, Rotate, Point) and system commands (Hold, Action), as summarized in Fig. 3.

- Task Commands
  - Point: indicates target locations or motion paths.
  - Pick/Place: refers to object grasping/releasing actions, mimicking human hand movements during physical interactions with objects.
  - Rotate: adjusts object orientation by tracking relative hand poses, mimicking how humans manually demonstrate rotation.
- System Commands
  - Hold: pauses programming to allow users to adjust their intentions (e.g., reorienting an object mid-task).
  - Action: resumes programming after a pause, signaling that the user is ready to input the next instruction.

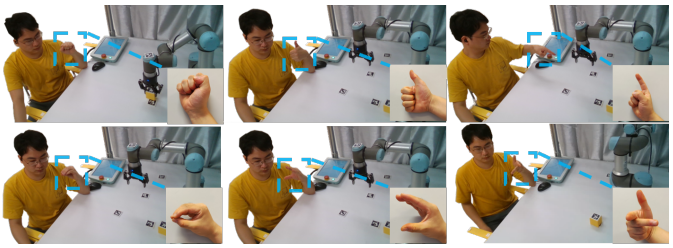

Fig. 3. Six gestures for robot programming commands. In the first row, there are three gestures: hold, action, and point. In the second row, the three gestures are pick, place, and rotate.

The design of Hold and Action gestures is inspired by human conversational dynamics. Just as people use pauses (e.g., raised hands) to halt discussions and verbal cues (e.g., "go ahead") to resume, these gestures provide non-technical end-users with natural control over programming workflows. Furthermore, this design ensures semantic alignment: each gesture's meaning (e.g., pick as a grasping motion) directly maps to its robotic function, reducing end-users' cognitive load. Additional gestures (e.g., Push/Pull) can also be integrated following the same alignment principle to enrich programming commands. Because robot programming relies on human-demonstrated gestures, prolonged or repetitive hand motions may cause muscle fatigue. To mitigate this problem, we propose intuitive gesture designs aligned with natural movements, together with a robust recognition network that is tolerant to variation.

C. Programming Methods

To enable end-users to program robots through body-language commands, we propose a method that integrates five core modules to translate detected human body language into executable robot actions. The system architecture is shown in Fig. 2 (green wire-frame), and the key design motivations are explained below.

1) Body-language Detection Model: Our detection module, based on the ConvLSTM network, processes 16 consecutive RGB frames (see Appendix B) to recognize dynamic gestures. Unlike single-frame approaches (e.g., YOLOv5 [33]), this temporally continuous model mitigates transient misclassification during gesture transitions [39]. For instance, a brief hand tremor during a "Pick-and-Place" sequence is filtered out by analyzing motion patterns across frames. This design mimics human perception, in which intent is inferred from sustained actions rather than isolated poses.

2) Location Programming: Humans commonly indicate location using their gaze and gestures. Drawing on this observation, we developed a network for programming the location command L based on both human gaze and gestures.

The architecture of the L programming network is shown in Fig. 4 and comprises three main blocks: a direction block, a fusion block, and a heatmap block. The direction block is responsible for estimating the direction of gaze or hand-pointing based on the input image, which includes the human's head and hand. The fusion block generates the direction field based on the gaze or hand-pointing direction, while the heatmap block estimates the location L within the direction field. This network offers two advantages for L command programming. First, it aligns with the human practice of using gaze and gestures to indicate a target location when it is within sight. Second, it provides a more accurate means of estimating the location by utilizing both gaze and hand-pointing inputs,

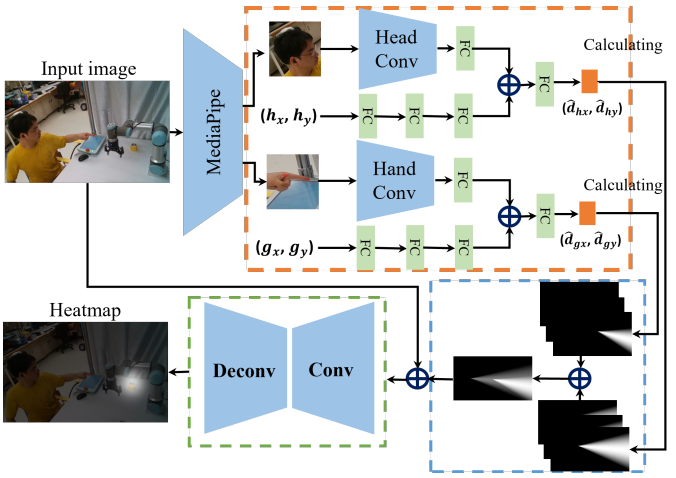

Fig. 4. Outline of the location L programming. There are three parts in this network, i.e., direction block (orange box), map fusion block (blue box), and heatmap block (green box).

as opposed to relying on just one input, such as gaze alone [35] or hand gestures alone [19].

Fig. 5. A description of the methodology for creating a gaze/hand-pointing direction field. The blue line represents the direction of the individual's gaze, while the green line indicates the direction of the hand-pointing gesture.

To establish the direction field (as depicted in Fig. 4), we utilize cones to delineate the field of view (FOV) and the field of pointing (FOP). The apex of the cone corresponds to the position of the head or the index fingertip. For the FOP, the index fingertip within the input image is denoted as $G = (g_x, g_y)$, and a given point $P_g = (p_{gx}, p_{gy})$ lies within the cone. Consequently, the likelihood that point P_g corresponds to location L depends on the pointing angle θ_g, as illustrated in Fig. 5. This probability is computed according to Eq. 2. The vector $G_g = (p_{gx} - g_x, p_{gy} - g_y)$ denotes the direction of the line GP_g, while $\hat{d}_g = (\hat{d_{gx}}, \hat{d_{gy}})$ signifies the anticipated pointing direction. Analogous to the FOP, we are also able to establish the FOV and subsequently ascertain the direction of the gaze field. To ensure that the target falls within the direction field, we generate multiple direction fields by varying the scale of the pointing angle $\theta_g = \{10°, 30°, 50°\}$ and gaze angle $\theta_h = \{30°, 60°, 90°\}$.

$$Pr_g = max(\frac{< G_g, \hat{d}_g >}{|G_g||\hat{d}_g|}, cos(\theta_g)). \tag{2}$$

The fusion direction field is a composite probability map (Pr_J, Eq. 3) derived from the integration of the two distinct direction fields (gaze and hand-pointing). Here, Pr_g denotes the probability distribution map of the field of pointing (FOP), while Pr_h signifies the probability distribution map of the field of view (FOV). The parameter α is assigned a value of 0.55.

$$\begin{cases} \hat{Pr_J} = \alpha Pr_g + (1-\alpha) Pr_h, \\ Pr_J = \frac{\hat{Pr_J} - min(\hat{Pr_J})}{max(\hat{Pr_J}) - min(\hat{Pr_J})}. \end{cases} \tag{3}$$

Following the generation of the fusion direction field, it is integrated with the original image. Subsequently, both are jointly input into the heatmap block to facilitate the regression of the heatmap. The heatmap block encompasses a Sigmoid function within its final layer to standardize the pixel values.

Typically, the pixel with the highest value in the heatmap may not correspond to any object within the original image. To solve this problem, we assume that L denotes the position of the object containing the highest value in the heatmap, as computed by Eq. 4. Here, P_{Oi} denotes the probability of the i-th target within the heatmap, which is determined as the mean probability of all pixels within the object (Appendix C). In practical programming scenarios, we mitigate the Midas Touch Problem by implementing a dwell time threshold and integrating multimodal inputs, thereby reducing false activations and enhancing the accuracy of location programming. Furthermore, when ambiguity arises in target selection under Eq. 4, we reduce the pointing and gaze angles to refine target localization.

$$\begin{cases} L_i = max(P_{O1}, P_{O2}, \ldots, P_{Oi}), \\ P_{Oi} = \sum_{j=1}^{n} P_{Oi}(x_j, y_j)/n. \end{cases} \tag{4}$$

3) Operation Programming: There are three basic gestures designed for operation programming, i.e., Pick, Place, and Rotate. After these commands are detected, module O translates them into corresponding robot actions. For instance, in the case of the pick or place instructions, the robot executes pretrained picking or placing skills. However, the rotate instruction poses a significantly greater challenge compared to other operation commands. This complexity stems from the requirement to accurately interpret the relationship between the robot's workspace coordinate system and its end-effector coordinate system, i.e., to precisely define the desired target pose within the robot's workspace coordinate system. However, this requirement is difficult for non-technical end-users to satisfy.

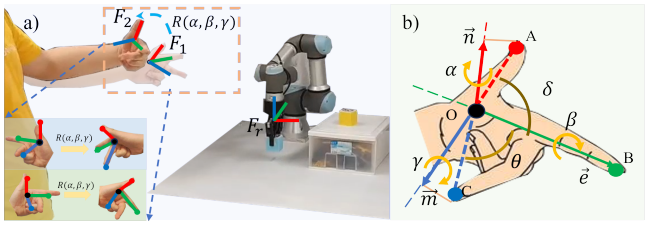

Fig. 6. Utilizing "Rotate" gesture to modify the robot's target pose. (a) shows that the end-user employs two "Rotate" gestures to program the robot with the rotation strategy $R(\alpha, \beta, \gamma)$. (b) illustrates how the gesture coordinate system F_i is calculated from four hand landmarks.

To reduce the difficulty of programming rotation instructions, we propose a method that uses relative coordinates to define a robot pose-adjustment strategy. Using this method, the end-user considers the robot's end-effector pose and specifies the rotation operation through intuitive hand movements, as

shown in Fig. 6. This approach avoids complex mathematical derivations and aligns with human spatial reasoning, which is beneficial for non-technical end-users. As depicted in Fig. 6 (a), the operator utilizes two gestures ("Rotate") to demonstrate the pose adjustment strategy $R(\alpha, \beta, \gamma)$. The coordinate frame F_1 of the first gesture represents the initial pose of the target, while the coordinate frame F_2 of the second gesture represents the desired target pose. Given that users focus on using the robot end-effector to adjust objects to target poses, we employ F_r (the coordinate system of the robotic end-effector) to represent the target pose within the robot's workspace.

After obtaining the gesture coordinate frames F_1 and F_2, we calculate the rotation matrix $R(\alpha, \beta, \gamma)$ from F_1 to F_2 (see Appendix D), as depicted in Eq. 5. The matrix $R_i = \{R_l, R_r\}$ is employed to represent the transformation relationship from the left/right-handed coordinates (R_l/R_r) to F_r.

$$R(\alpha, \beta, \gamma) = R_{F_2}^{F_1} = R_i F_2 \cdot (R_i F_1)^{-1}. \tag{5}$$

4) Motion Programming: The motion component M defines the trajectory for the robot to move between two locations while adhering to user preferences and environmental constraints. Traditional path-planning algorithms (e.g., A*) often fail to incorporate nuanced human preferences, such as avoiding hazards that robots cannot perceive or adhering to specific motion styles. To address this problem, we propose an intuitive programming method, described in Algorithm 1, in which users sketch desired trajectories in the air using the "Point" gesture. These trajectories are then translated into executable robot commands through a optimization framework based on a genetic-algorithm (GA).

Algorithm 1: Motion M Programming

input : several images, two locations L_1, L_2
output: The motion strategy M_s

1 **for** $i \leftarrow 1$ **to** l **do**
2 $(x_p, y_p) \leftarrow$ `FPImage(`*Image*`)` ;
3 $(z_c) \leftarrow$ `FindDepth(`*Image*(x_p, y_p)`)` ;
4 $(x_c, y_c, z_c) \leftarrow$ `CalPose(`x_p, y_p, z_c`)` ;
5 **end**
6 $M \leftarrow$`Filter(`$\{(x_c, y_c, z_c)\}$`)` ;
7 $M_c \leftarrow$ `GA(`$\{x_c, y_c, z_c\}$`)` ;
8 **for** $i \leftarrow 1$ **to** $len(M_c)$ **do**
9 $M_r\{i\} \leftarrow$ `CoorTrans(`$R_c^r, M_c\{i\}$`)` ;
10 $M_r\{i\} \leftarrow$ `ScaleTrans(`$M_r\{i\}, L_1, L_2$`)` ;
11 $M_r\{i\} \leftarrow$ `RotTrans(`$M_r\{i\}, L_1, L_2$`)`;
12 $M_r\{i\} \leftarrow$ `DispTrans(`$M_r\{i\}, L_1, L_2$`)` ;
13 **end**
14 $M_s \leftarrow$ `CheckColl(`M_r`)`;

Algorithm 1 outlines the workflow of M programming, which can be segmented into four distinct parts. In the initial segment (lines 1 to 6), we extract the 3D coordinates of the end-user's index fingertip in the camera coordinate system from the image captured by the camera. Subsequently, a genetic algorithm (GA, line 7) is used to extract multiple key points from the trajectory demonstrated by the end-user, which is then recorded as motion strategy M_c. Following GA, spatial operations are employed to map the motion strategy M_c from the camera coordinate system to M_r in the robot coordinate system (lines 8 to 13). Here, M_r represents the robot motion strategy specified by the user in the robot coordinate system. Finally, the algorithm checks for potential collision risks along the path (line 14) (see Appendix F). After these four steps, the motion strategy M_s is generated for the robot.

$$F = \frac{1}{|(n - N_g)|} + \sum_{i}^{N_g} \sum_{j}^{N_c} |(P_j - P_{n_i})| + \sum_{i}^{N_g} |C_{n_i}|. \tag{6}$$

In Algorithm 1, the GA (see Appendix E) is equipped with a carefully designed fitness function. The fitness (F) is computed based on Eq. 6, which includes three components: number, distribution, and curvature. Here, N_g denotes the desired number of dominant genes, and n represents the actual number of dominant genes. Additionally, P_j signifies the position of a point on the motion strategy M, while P_{ni} denotes the position of dominant genes. Furthermore, C_{n_i} represents the curvatures at point n_i in the original strategy M. Finally, the GA with a well-designed fitness can provide a simplified motion strategy by extracting key features from the user's demonstration.

5) Robot Brain: To synthesize the detected motion commands L, O, and M into executable robot actions, we introduce an LLM-based (GPT-4o) reasoning module, referred to as the robot brain. This reasoning module is introduced to handle differences in task understanding and programming habits among end-users. Different users may adopt different task sequences and gesture inputs during programming. For example, a standard programming sequence for a pick-and-place task is typically L, O, M, L, O, but some users may prefer to use a sequence such as L, L, O, O, M. These differences often stem from varying task cognition or personal behavioral habits. To ensure robots can handle these differences, we designed the robot brain module, which can infer the inherent relationships between user instructions and automatically adjust the task execution sequence to meet the desired outcomes.

The core of this reasoning module is the LLM, which can comprehend and handle complex task semantics, infer the dependencies between different commands and significantly improve the robot's adaptability to user inputs. To enable the LLM to function as a robot brain, we design a prompt (see Appendix G) that guides it to reason about the relationships between human commands and generate executable interaction-task sequences I_i. After obtaining the executable sequence I_i, the robot can perform the task based on its pretrained skills.

IV. Evaluation Results

In this section, we present a series of demonstrations and evaluations to illustrate the functionality and effectiveness of our robot programming method. First, we evaluate our body-language detection network, followed by separate demonstrations of the performance of each programming module, namely L, O, and M. Finally, we conduct two experiments on different

robot platforms to demonstrate the effectiveness of our method for programming long-horizon HRI tasks.

A. Body-language Detection

In order to train the body-language detection network, we curated a dataset comprising user-friendly gestures that corresponded to robot programming commands outlined in Section III-B. This dataset includes six gesture types, with more than 6000 images for each gesture (see Appendix A). The hyperparameters for this model were configured as follows: batch size (128), learning rate (0.01), and weight decay (0.0001).

TABLE I
COMPARISON RESULTS ON OUR DATASET FOR SEVERAL GESTURE DETECTION METHODS

Method	Precision	Recall	F1-score
CNN [32]	0.822	0.811	0.812
YOLOv5s [33]	0.856	0.881	0.862
Ren et al. [40]	0.862	0.873	0.865
FC-LSTM [41]	0.901	0.924	0.915
PredRNN [42]	0.934	0.917	0.921
Ours	0.946	0.959	0.951

We conducted a comparative evaluation of our method against other state-of-the-art approaches using three metrics: precision, recall, and F1-score. The results of this comparison are presented in Table I. Despite the strong performance of YOLOv5s in gesture detection and classification, its adaptability to the application scenario considered in this paper is limited. This limitation arises from its reliance on single-frame image-based detection, which often leads to errors during robot programming. For instance, when the end-user transitions from the Pick gesture to another gesture, the YOLOv5s model may erroneously classify it as Place.

To mitigate this issue, we introduced a detection network based on the ConvLSTM model, which leverages a sequence of 16 consecutive frames. Comparative evaluation against state-of-the-art baselines demonstrates the effectiveness of our model for gesture detection, particularly in continuous body-language detection scenarios.

B. Location Programming

Existing object-detection datasets rely solely on either human gaze [36] or gestures [19]. To address this limitation, we developed a new object-detection dataset that includes both human gaze and gestures, thereby supporting network training and testing. Our gesture-gaze dataset comprises more than 6000 images collected across four distinct scenes, with the target location in each image manually labeled (see Appendix A). To evaluate the discrepancy between the ground-truth location and the predicted location, we used four metrics: area under curve (AUC) [37], Euclidean distance (Dist) [36], minimum Euclidean distance (MDist) [36], and angular error (Ang) [36]. We then selected several state-of-the-art methods for gaze detection and pointing-direction estimation and compared them with our proposed method. The results are presented in Table II.

TABLE II
EVALUATION RESULTS ON OUR DATASET

Method	AUC	Dist	Mdist	Ang
Fixed bias [36]	0.598	0.412	0.286	53.0°
SVM+shift grid [36]	0.791	0.251	0.186	43.2°
Judd et al. [37]	0.692	0.324	0.254	48.4°
SalGAN [38]	0.814	0.203	0.198	21.7°
Recasens et al. [36]	0.842	0.164	0.124	24.0°
GazeFollowing [35]	0.916	0.143	0.095	19.2°
GaTector [23]	0.936	0.107	0.085	15.7°
MediaPipe [31]	0.862	–	–	24.0°
VNect [43]	0.716	–	–	30.2°
OpenPose [34]	0.831	–	–	16.1°
PointingNet [19]	0.942	0.084	0.072	8.7°
Ours (only-gaze)	0.912	0.159	0.091	18.4°
Ours (only-hand)	0.939	0.093	0.071	9.1°
Ours (whole)	**0.953**	**0.077**	**0.068**	**6.9°**

The results of our method (only-gaze/only-hand) were obtained by deactivating the gesture/gaze channel and replacing the corresponding direction field with an all-black field. Given that our method (only-gaze) exclusively relies on human gaze, the results closely align with those of GazeFollowing [35], and are inferior to those of GaTector [23]. Notably, because gestures are comparatively easier to detect and pointing directions are more directly indicative of target position, our hand-based method outperforms the gaze-based approach. After integrating the two channels, the results in Table II show that our full model outperforms all baselines across all evaluation metrics, indicating a substantial overall improvement.

The prediction results are presented in Fig. 7. The first row illustrates the gaze/gesture direction, while the second row depicts the corresponding heatmaps. The third row displays the estimated target location, which is determined using Eq. 4 in conjunction with the heatmap.

Fig. 7. Some estimation results during the testing process for location L programming. The first row illustrates the estimated gaze/hand-pointing direction alongside the ground truth gaze/hand-pointing direction. The second row shows the heatmaps generated from the first row. The third row shows the target estimated based on the heatmap.

C. Operation Programming

Fig. 8 presents the evaluation results of the pose-adjustment strategy for both hands. In the experimental setup, a Rotate gesture was utilized to adjust the target pose from F_1 to F_2, as illustrated in Fig. 6. Two angles within the range of $[-\pi/2, \pi/2]$ were selected for this experiment, both under clockwise rotation. For each rotation case, such as

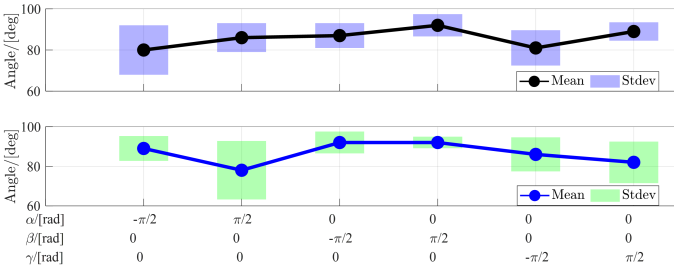

Fig. 8. Evaluation results of the pose-adjusting strategy. The initial row corresponds to the results for the left hand, while the subsequent row pertains to the outcomes for the right hand. The rotation angle is denoted as an absolute value.

$\alpha = \pi/2, \beta = 0, \gamma = 0$, a total of 15 attempts were conducted. The findings indicate that the rotation strategy in the β direction outperforms that in the other two directions. This observation can be attributed to the construction of the gesture coordinate frame F_i based on the direction of $\vec{OB}$. However, programming robot rotation accurately is difficult for humans, as evidenced by the large standard deviations observed across rotation conditions in Fig. 6. The main reason for this challenge is that humans tend to perceive space in relative poses rather than precise absolute poses. To address this issue, an incremental rotation strategy was employed. This strategy enables the robot to rotate the target by a specific angle (e.g., $\theta = 90°$) in the designated direction after receiving the Rotate instruction.

D. Motion Programming

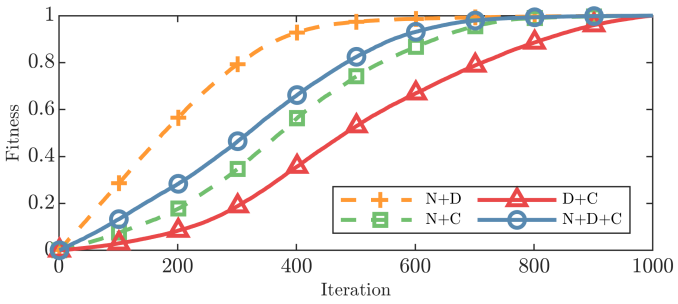

Fig. 9. Trends of four fitness functions during the GA process "N", "D", and "C" refer to the number of individuals, the distribution of individuals, and the curvatures of individuals, respectively.

For motion programming M, as outlined in Algorithm 1, the key issue is the design of the GA fitness function. Fig. 9 illustrates four fitness curves corresponding to different fitness functions, derived from processing the circular motion strategy shown in Fig. 10. In this experiment, we set the number of dominant genes N_g to 10 and selected the top 100 individuals with the highest fitness to progress to the next generation.

The outcomes of the GA are shown in Fig. 10. In this study, we selected a fitness function based on the number, distribution, and curvature of individuals, denoted as "N+D+C". Compared with alternative designs, this fitness function generates fewer waypoints (in contrast to Fig. 10-c), distributes them more sparsely along the curve (in contrast

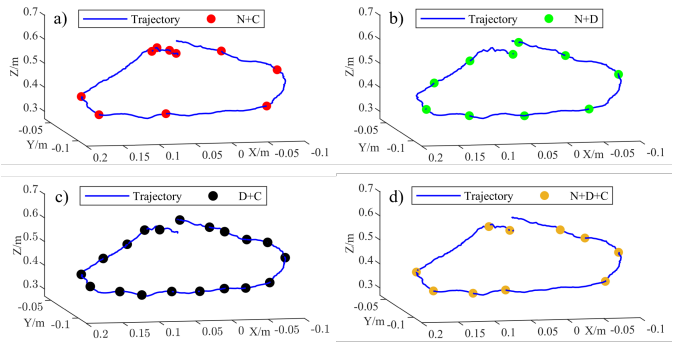

Fig. 10. Selected trajectory points generated by the GA under different fitness functions. The meanings of the labels (i.e., N, D, and C) are the same as those in Fig. 9.

to Fig. 10-a), and better preserves meaningful characteristics of the original motion strategy, such as regions with significant curvature. Although the convergence rate of the "N+D+C" fitness function is lower than that of "N+D" (see Fig. 9), it remains acceptable compared with alternative fitness designs.

E. Case study

In this section, we present two HRI case studies with different robot platforms to demonstrate the application of our methodology for long-horizon robot programming tasks. In the first case, the end-user programs the UR arms to manipulate distant cubes and relocate them to nearby positions. In the second case, the end-user programs the humanoid robot GR1 to perform desk-tidying tasks. Initially, the end-user adopts a task-decomposition strategy, treating the movement of each target as a distinct subtask. The programming process is then carried out sequentially at the subtask level, with each subtask completed before proceeding to the next.

Fig. 11 illustrates the first case, which consists of two subtasks. Each subtask involves two procedures: human programming and robot action. In the tasks, the end-user programs two robot arms to relocate a remote cube to a specified target location. Due to the workspace limitations of the robot arms, the end-user first programs the left robot arm to transfer the cube to a position within the workspace of the right robot arm. The right arm is then programmed to continue the relocation task.

During the programming process, the end-user specifies the three elements of I_i for the UR arm to describe the subtask. As shown in Fig. 11 (blue block, first row), the programming sequence includes the initial location L_1 (sequence 1), the motion M_s (sequences $1-4$), the end location L_2 (sequence 4), and the operations O (sequences $5-6$) including Pick (sequence 5) and Place (sequence 6), i.e., $I_i = \{L_1, M_s, L_2, O_{1,2}\}$. Since the Hold and Action gestures are used to control the programming process, they are not shown in the programming sequence in Fig. 11.

After the programming process is completed, the robot executes the subtask. As depicted in Fig. 11 (green block, second row), the UR arm first moves from its initial position (sequence 1) to the target location L_1 (sequence 2). Subsequently, upon grasping the cube (sequence 3), it transports the cube to the

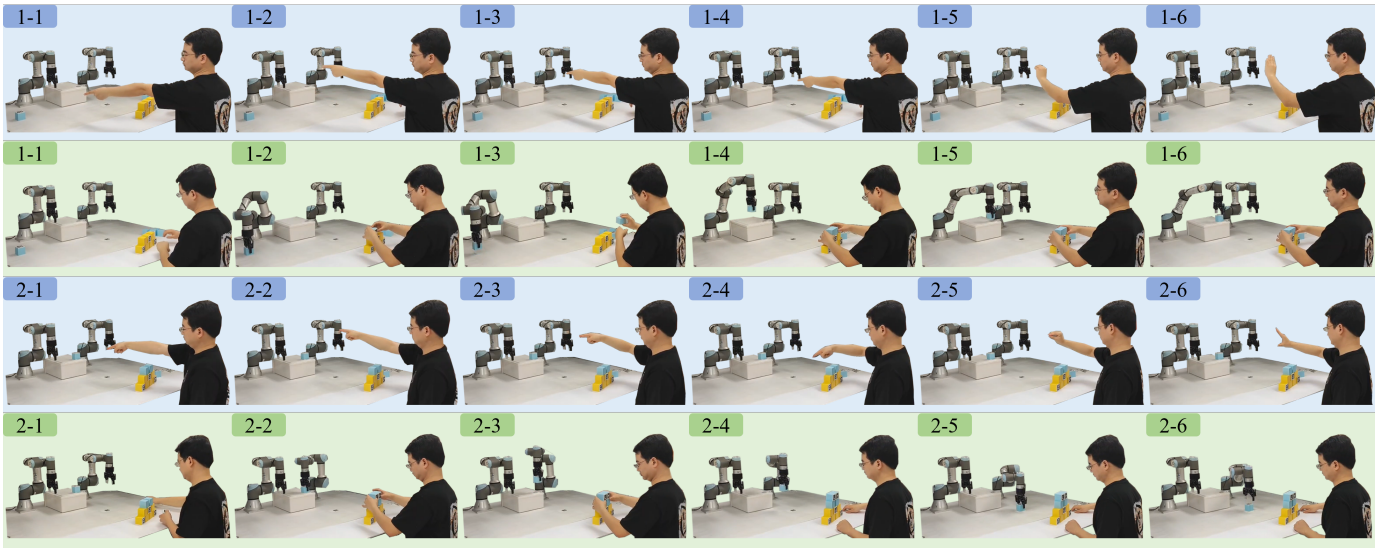

Fig. 11. Robot programming procedures for two subtasks of the "Playing the Building Blocks Together" task. For each subtask, the blue blocks denote the human programming stage, in which the user specifies the task elements $\{L, M, O\}$ through body language, whereas the green blocks denote the subsequent robot execution stage. The numbered snapshots (e.g., 2-1) indicate the temporal progression of the task: the first digit denotes the subtask index, while the second digit denotes the action order of the end-user or the robot within that subtask. The sequence further shows that robot execution begins only after the corresponding human programming process is completed.

end location L_2 (sequence 6) in accordance with the motion strategy M_s (sequences $3-6$). Finally, the UR arm places the cube at the end location L_2 (sequence 6) and returns to its initial position (sequence 1). To complete the overall task, end-users repeat this interaction process until all subtasks are completed. As shown in Fig. 11, robot execution always occurs after user programming: the user first inputs the instructions through body language, and the robot then performs the corresponding actions after receiving the complete subtask instructions.

For the second task illustrated in Fig. 12, the programming framework is structurally similar to that of the first task. The key difference lies in the implementation: the humanoid robot detects the user's body language through its onboard vision system (equipped with RealSense D435i cameras) and localizes target objects via a depth-estimation algorithm, instead of using the QR-code-based localization method adopted in the first task.

F. End-user study

To rigorously validate the system's usability for non-technical end-users, we conducted a comprehensive end-user study involving 8 participants (5 males, 3 females; aged 18–55) with no prior robotics experience. Participants were invited from diverse backgrounds (e.g., office workers, students, and homemakers) to ensure generalizability. As a reference group, five professionals (4 males, 1 female; aged 24–32) also participated in this study. This study was divided into two stages to simulate real-world robot programming scenarios of increasing complexity:

- Stage 1 (Basic task): Participants were asked to program all task elements I_i (i.e., L, M, and O) for both the robot arms and the humanoid robot. In this stage, we evaluated users' mastery of basic gesture commands (e.g., pick).
- Stage 2 (Long-horizon task): Participants were required to complete the four-object rearrangement task, which consists of four basic tasks, using both the robot arms and the humanoid robot. In this stage, we evaluated end-users' performance in complex task decomposition and multi-step programming.

To evaluate end-user performance and the usability of our framework, several metrics were introduced in this study. Task success rate (TSR) refers to the ratio of successful task completions to total attempts. The average programming time (Avg. PT) refers to the duration required for users to generate complete robot instructions of an entire task, including gesture input, path planning, and error correction. Error Correction (EC) per task is the average number of times end-users paused and adjusted erroneous steps during the robot programming process. User Satisfaction (USatis) reflects users' subjective ratings of the system's ease of use and interaction fluency. The USatis score was collected on a 5-point scale (score 1 means very dissatisfied, 5 means very satisfied). For each evaluation metric, we adopted an equivalence-testing framework to assess whether the performance of end-users was practically equivalent to that of experts when using our method. For each metric, the null hypothesis stated that the performance difference between the end-user group and the expert group was greater than or equal to the predefined equivalence margin, whereas the alternative hypothesis stated that the difference was smaller than the corresponding margin, indicating practical equivalence. The equivalence margins were set to $\Delta = 5\%$ for TSR, $\Delta = 10s$ for Avg. PT in Study 1 and $\Delta = 40s$ in Study 2, $\Delta = 1$ for EC in Study 1 and $\Delta = 4$ in Study 2, and $\Delta = 0.5$ for USatis. These margins were selected to reflect the maximum practically negligible differences for each metric, considering the task scale, completion time, and rating resolution used in the two studies. Then, we conducted

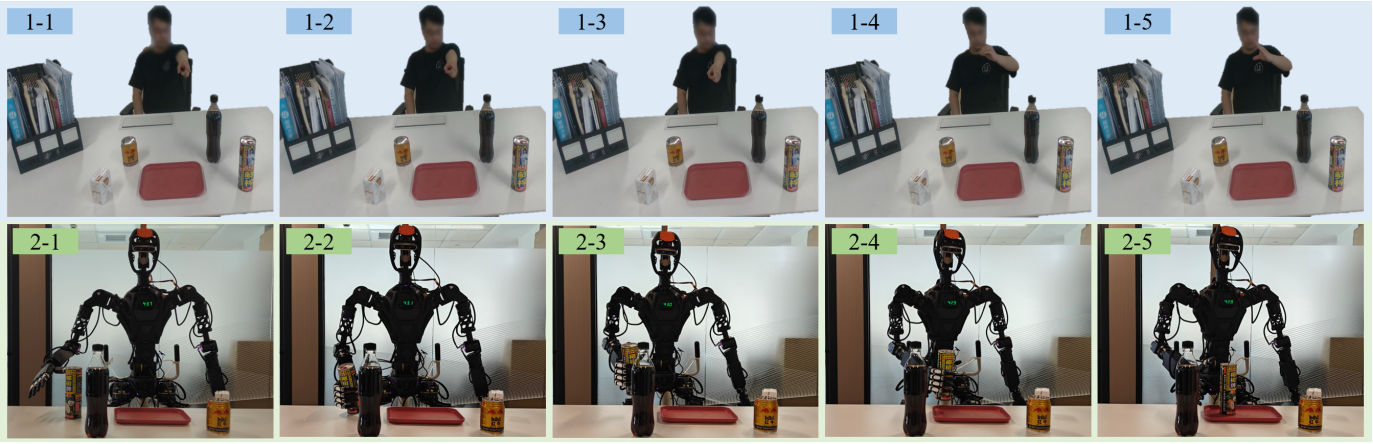

Fig. 12. Robot programming procedures for the desk-tidying task using the humanoid robot. The blue blocks denote the human programming stage, in which the user specifies the task elements $\{L, M, O\}$ through body language, whereas the green blocks denote the subsequent robot execution stage. The numbered snapshots (e.g., 2-1) indicate the temporal progression of each subtask, with the first digit denoting the subtask index and the second digit denoting the action order. Both body-language interpretation and target localization are performed by the humanoid robot's onboard vision system.

TABLE III
USER-STUDY RESULTS FOR THE TWO-STAGE EXPERIMENT.

Scenarios	Metrics	End-users	Experts	p_{eq}	90% CI
Stage 1	TSR(%)	96.0	100.0	<0.01	[-4.6, -3.4]
	Avg. PT(s)	35.5	29.0	<0.05	[3.0, 9.9]
	EC	0.4	0.0	<0.05	[0.1, 0.6]
	USatis	4.1	4.4	<0.01	[-0.4, 0.1]
Stage 2	TSR(%)	83.0	100.0	>0.05	[-19.2, -14.7]
	Avg. PT(s)	351.0	253.0	>0.05	[84.9, 111.2]
	EC	3.4	0.2	<0.05	[2.4, 3.9]
	USatis	3.4	4.2	>0.05	[-1.2, -0.4]

bootstrap-based equivalence tests at a significance level of 0.05 for all user-study metrics. For each metric, equivalence was established when the bootstrap confidence interval of the mean difference between the end-user group and the expert group fell entirely within the corresponding predefined equivalence interval. The reported p-values (p_{eq}) correspond to bootstrap-based equivalence tests rather than conventional difference tests.

The results in Table III show that, in Stage 1, end-users achieved performance that was practically comparable to that of experts on several metrics. Specifically, the bootstrap 90% confidence intervals for TSR, Avg. PT, EC, and USatis were fully contained within their corresponding predefined equivalence bounds, and the null hypothesis of non-equivalence was therefore rejected for these metrics in Stage 1. These findings suggest that the proposed framework enables non-technical end-users to achieve performance comparable to that of experts in basic tasks, although the interaction experience can still be further improved. In Stage 2, practical equivalence between end-users and experts was not established for most of the evaluation metrics. By contrast, the result for EC remained within the equivalence range. The bootstrap 90% confidence intervals of TSR, Avg. PT, and USatis all exceed their corresponding predefined equivalence bounds, indicating that the performance gap between the two groups became substantially larger in long-horizon tasks. This result suggests that, while the proposed framework is effective for basic robot programming, its robustness and usability remain limited when end-users are required to perform complex task decomposition and multi-step programming. Overall, these results indicate that the performance differences between the two groups exceeded the predefined equivalence bounds in long-horizon tasks.

V. CONCLUSION

This paper presents a vision-driven robot programming method that enables non-technical end-users to specify robotic tasks through intuitive body-language commands, such as gaze and gestures. By decomposing complex interactions into three components (L, O, and M) and aligning them with predefined body-language commands, our framework allows end-users to program robotic tasks without coding skills or robotics knowledge. The integration of ConvLSTM-based gesture recognition and a fusion network for gaze-hand direction estimation achieves robust command detection, while the LLM-based reasoning module enhances adaptability to diverse user programming sequences. Experimental evaluations demonstrate the system's effectiveness on basic tasks across different robot platforms, with non-technical users achieving a 96% success rate in basic operations. However, performance gaps remain in long-horizon tasks, highlighting challenges in multi-step coordination.

Despite its advantages, the current method has several limitations. First, the user study involved a limited sample size (8 non-technical participants and 5 experts), restricting generalizability to broader demographics, such as elderly or disabled users. Second, the vision-driven model introduces latency ($\geq$ 120 ms), which may affect real-time responsiveness in fast-paced interactions. Lastly, the current gesture vocabulary supports only six commands, which limits the scalability of the framework for more complex operations. Our future work will focus on two directions: expanding gesture semantics through multimodal integration (e.g., voice feedback) and validating the framework in larger-scale trials with diverse user groups. Addressing these limitations will

enhance the system's practicality for real-world human-robot collaboration.

REFERENCES

[1] L. Bärmann, R. Kartmann, F.n Peller-Konrad, A. Waibel, and T. Asfour. Incremental learning of humanoid robot behavior from natural interaction and large language models. *arXiv preprint arXiv:2309.04316*, 2023.

[2] M. Mahmud, M. Abidin, S. Buyamin, A. Emmanuel, and H. Hasan. Multi-objective route planning for underwater cleaning robot in water reservoir tank. *Journal of Intelligent & Robotic Systems*, 101:1–16, 2021.

[3] T. Shimmura, R. Ichikari, T. Okuma, H. Ito, K. Okada, and T. Nonaka. Service robot introduction to a restaurant enhances both labor productivity and service quality. *Procedia CIRP*, 88:589–594, 2020.

[4] Z. Chen, Y. Nakamura, and H. Ishiguro. Android as a receptionist in a shopping mall using inverse reinforcement learning. *IEEE Robotics and Automation Letters*, 7(3):7091–7098, 2022.

[5] G. Ajaykumar, M. Steele, and C. Huang. A survey on end-user robot programming. *ACM Computing Surveys (CSUR)*, 54(8):1–36, 2021.

[6] S. Blankemeyer, R. Wiemann, L. Posniak, C. Pregizer, and A. Raatz. Intuitive robot programming using augmented reality. *Procedia CIRP*, 76:155–160, 2018.

[7] F. Brizzi, L. Peppoloni, A. Graziano, E. Stefano, C. Avizzano, and E. Ruffaldi. Effects of augmented reality on the performance of teleoperated industrial assembly tasks in a robotic embodiment. *IEEE Transactions on Human-Machine Systems*, 48(2):197–206, 2018.

[8] M. Walter, M. Antone, E. Chuangsuwanich, et al. A situationally aware voice-commandable robotic forklift working alongside people in unstructured outdoor environments. *Journal of Field Robotics*, 32(4):590–628, 2015.

[9] Z. Wang, H. Wang, H. Yu, and F. Lu. Interaction with gaze, gesture, and speech in a flexibly configurable augmented reality system. *IEEE Transactions on Human-Machine Systems*, 51(5):524–534, 2021.

[10] S. Lathuilière, B. Massé, P. Mesejo, and R. Horaud. Neural network based reinforcement learning for audio–visual gaze control in human–robot interaction. *Pattern Recognition Letters*, 118:61–71, 2019.

[11] J. Lee, Y. Kim, S. An, and S. Bae. Robot telekinesis: application of a unimanual and bimanual object manipulation technique to robot control. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 9866–9872. IEEE, 2020.

[12] C. Quintero, S.h Li, M. Pan, W. Chan, et al. Robot programming through augmented trajectories in augmented reality. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1838–1844. IEEE, 2018.

[13] M. Ostanin, R. Yagfarov, and A. Klimchik. Interactive robots control using mixed reality. *IFAC-PapersOnLine*, 52(13):695–700, 2019. 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019.

[14] P. Vivek K., Boris I., Bastian G., Bernd B., and Dirk S. Mobile manipulator control through gesture recognition using imus and online lazy neighborhood graph search. *Acta IMEKO*, 8(4):3–8, 2019.

[15] A. Padmanabha, Q. Wang, D. Han, J. Diyora, K. Kacker, H. Khalid, L. Chen, C. Majidi, and Z. Erickson. Hat: Head-worn assistive teleoperation of mobile manipulators. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12542–12548, 2023.

[16] S. Bianco, P. Napoletano, A. Raimondi, and M. Rima. U-wear: User recognition on wearable devices through arm gesture. *IEEE Transactions on Human-Machine Systems*, 52(4):713–724, 2022.

[17] Z. Jiang, Y. Xie, J. Li, Y. Yuan, Y. Zhu, and Y. Zhu. Harmon: Whole-body motion generation of humanoid robots from language descriptions. In *CoRL 2024 Workshop CoRoboLearn: Advancing Learning for Human-Centered Collaborative Robots*.

[18] Z. Zhang, A. Lin, C. Wai Wong, X. Chu, Q. Dou, and K. Au. Interactive navigation in environments with traversable obstacles using large language and vision-language models. *arXiv preprint arXiv:2310.08873*, 2023.

[19] E. Bamani, E. Nissinman, L. Koenigsberg, I. Meir, Y. Matalon, and A. Sintov. Recognition and estimation of human finger pointing with an rgb camera for robot directive. *arXiv preprint arXiv:2307.02949*, 2023.

[20] C. Cuan, E. Lee, E. Fisher, A. Francis, L. Takayama, T. Zhang, A. Toshev, and S. Pirk. Gesture2path: Imitation learning for gesture-aware navigation. *arXiv preprint arXiv:2209.09375*, 2022.

[21] O. Mazhar, B. Navarro, S. Ramdani, R. Passama, and A. Cherubini. A real-time human-robot interaction framework with robust background invariant hand gesture detection. *Robotics and Computer-Integrated Manufacturing*, 60:34–48, 2019.

[22] A. Fischer-Janzen, T. Wendt, and K. Van Laerhoven. A scoping review of gaze and eye tracking-based control methods for assistive robotic arms. *Frontiers in Robotics and AI*, 11:1326670, 2024.

[23] B. Wang, T. Hu, B. Li, X. Chen, and Z. Zhang. Gatector: A unified framework for gaze object prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19588–19597, 2022.

[24] D. Tu, X. Min, H. Duan, G. Guo, G. Zhai, and W. Shen. End-to-end human-gaze-target detection with transformers. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2192–2200. IEEE, 2022.

[25] L. Wang and S. Li. Wheelchair-centered omnidirectional gaze-point estimation in the wild. *IEEE Transactions on Human-Machine Systems*, 53(3):466–478, 2023.

[26] A. Shafti, P. Orlov, and A. Faisal. Gaze-based, context-aware robotic system for assisted reaching and grasping. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 863–869. IEEE, 2019.

[27] O Kuzmicheva and A Gräser. Gaze gesture-based human robot interface. *Technische Unterstützungssysteme, die die Menschen wirklich wollen*, page 339, 2016.

[28] V. Krishna Sharma, K. Saluja, V. Mollyn, and P. Biswas. Eye gaze controlled robotic arm for persons with severe speech and motor impairment. In *ACM symposium on eye tracking research and applications*, pages 1–9, 2020.

[29] L. Yuan, C. Reardon, G. Warnell, and G. Loianno. Human gaze-driven spatial tasking of an autonomous mav. *IEEE Robotics and Automation Letters*, 4(2):1343–1350, 2019.

[30] S. Yeamkuan, K. Chamnongthai, and W. Pichitwong. A 3d point-of-intention estimation method using multimodal fusion of hand pointing, eye gaze and depth sensing for collaborative robots. *IEEE Sensors Journal*, 22(3):2700–2710, 2021.

[31] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, et al. Mediapipe: A framework for perceiving and processing reality. In *Third workshop on computer vision for AR/VR at IEEE computer vision and pattern recognition (CVPR)*, volume 2019, 2019.

[32] G. Li, H. Tang, Y. Sun, J. Kong, G. Jiang, D. Jiang, B. Tao, S. Xu, and H. Liu. Hand gesture recognition based on convolution neural network. *Cluster Computing*, 22:2719–2729, 2019.

[33] L. Ling, J. Tao, and G. Wu. Research on gesture recognition based on yolov5. In *2021 33rd Chinese Control and Decision Conference (CCDC)*, pages 801–806. IEEE, 2021.

[34] D. Osokin. Real-time 2d multi-person pose estimation on cpu: Lightweight openpose. *arXiv preprint arXiv:1811.12004*, 2018.

[35] D. Lian, Z. Yu, and S. Gao. Believe it or not, we know what you are looking at! In *Asian Conference on Computer Vision*, pages 35–50. Springer, 2018.

[36] A. Recasens, A. Khosla, C. Vondrick, and A. Torralba. Where are they looking? *Advances in neural information processing systems*, 28, 2015.

[37] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *2009 IEEE 12th international conference on computer vision*, pages 2106–2113. IEEE, 2009.

[38] J. Pan, C. Ferrer, K. McGuinness, N. O'Connor, J. Torres, E. Sayrol, and X. Giro-i Nieto. Salgan: Visual saliency prediction with generative adversarial networks. *arXiv preprint arXiv:1701.01081*, 2017.

[39] L. Zhang, G. Zhu, L. Mei, P. Shen, S. Shah, and M. Bennamoun. Attention in convolutional lstm for gesture recognition. *Advances in neural information processing systems*, 31, 2018.

[40] S. Yan, Y. Xia, J. Smith, W. Lu, B. Zhang, et al. Multiscale convolutional neural networks for hand detection. *Applied Computational Intelligence and Soft Computing*, 2017, 2017.

[41] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852. PMLR, 2015.

[42] Y. Wang, M. Long, J. Wang, Z. Gao, and P. Yu. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. *Advances in neural information processing systems*, 30, 2017.

[43] D. Mehta, S. Sridhar, O. Sotnychenko, et al. Vnect: Real-time 3d human pose estimation with a single rgb camera. *Acm transactions on graphics (tog)*, 36(4):1–14, 2017.