Modular Customizable ROS-Based Framework for Rapid Development of Social Robots

1st Mahta Akhyani
dept. Electrical and Computer Engineering
University of Tehran
Tehran, Iran
Mahtaakhyani@gmail.com

2nd Hadi Moradi Sabzevar
dept. Electrical and Computer Engineering
University of Tehran
Tehran, Iran
moradih@ut.ac.ir

***Abstract*— Social robot development presents challenges in control, perception, navigation, and interaction design that impede rapid prototyping. This paper presents a modular architecture based on the Robot Operating System (ROS) to accelerate prototyping. Key technical contributions include 1) reusable nodes for core capabilities; 2) web interfaces for remote monitoring/control; 3) a framework for scripting interactions; 4) backend database integration; and 5) standardized ROS interfaces enabling customization. These capabilities provide essential abstractions to lower programming barriers and enhance reusability while maintaining customizability. Despite different hardware, the same logic and control were reused, validating the benefits of modularity and configurability for interactive development.**

Keywords—Human-Robot Interaction, Framework, Open Source, Rapid Prototyping, Social Robots, ROS

I. Introduction

The Robot Operating System (ROS) has become the standard platform for robot software development across industrial, personal, and social applications [1,2]. While facilitating sensor integration and behavior implementation [3], programming complex social robots in ROS remains challenging [4-6]. Most research relies on pre-built robots with limited capabilities [7-9]. Community contributions are fragmented rather than consolidated into cohesive platforms [10,11]. That is why the need for ROS-based social robot frameworks with essential capabilities and interfaces to accelerate development has been addressed [12-15].

Unfortunately, to the best of our knowledge, there has not been any work addressing this need. Consequently, in this paper, we present a modular ROS platform to streamline social robot prototyping. The customizable architecture provides web-based control, reusable ROS nodes, behavior scripting, database integration, and standardized interfaces between modules. These capabilities can lower programming barriers and reduce development costs while maintaining modularity for extension.

A set of ROS packages has been developed, with each package containing modules to handle various aspects of human-robot interaction within this platform. For example, for the computer vision and face detection component, the Python MediaPipe library from Google was utilized and extended such that its modules can run as ROS services, capturing camera frames and returning lists of facial landmark coordinates. These coordinates are then taken in a subsequent expanded package and added to the input camera image frame, converted to a standard ROS message, and passed along to continue through the framework pipeline.

In summary, modular ROS packages leveraging external libraries like MediaPipe have been created to enable robot capabilities like facial recognition, and these have been integrated into the overall ROS architecture to enable human-robot interaction.

A. Prior work

Many social robots, such as Pepper [1] and Nao [2], have proprietary integrated software stacks that constrain the range of behaviors and use cases. Tiago [3] uses a ROS-based modular robot software framework with a focus more on mobility rather than social capabilities. Based on our search on the research in this area, there has been limited work on customizable ROS-based platforms tailored for social robotics. A list of these platforms that have been developed to enable social robot capabilities is presented in Table 1. Unfortunately, most of these platforms offer limited customizability and modularity, which have been explained in the following.

This work enables the integration of ROS with Django through native Python APIs, providing a generalizable web interface for HRI lacking in previous ROS packages. It facilitates extensibility through modular packages that leverage state-of-the-art solutions like MediaPipe while adding new capabilities. This differs from other HRI systems like AMIRO, EVA, and Furhat which utilize alternative frameworks, interfaces, and robot platforms.

Other projects have enabled Django integration with specific robots, but a generalized interface was not created. JSON-based approaches like the ROSbridge Suite contrast the native Python APIs used here. Platforms like Misty, RobWork, and Fetch provide software development kits but lack readily extensible HRI capabilities. Systems including PEBBLES, PR2, and HARMONI focus on custom hardware rather than integrative software frameworks.

In summary, this work provides a novel contribution through its generalizable web interface for ROS HRI based on native Python APIs. It balances leveraging existing state-of-the-art libraries like MediaPipe with adding new functionalities for social robots. The integrative framework differs from specialized, non-extensible platforms and interfaces in other projects. This enables building

TABLE 1: COMPARISON OF PLATFORMS FOR DEVELOPING SOCIAL ROBOTS WITH THE PROPOSED PLATFORM

Platform	**Description**	**Key Features**	**Contribution**	**Open Source**	**ROS-based**	**Specific to social robots**	**Comparison to SROSP**
AMIRO [2]	Framework for developing social robots deployed on Pepper robot	Dialog system, emotion recognition, navigation, Pepper API integration	Deployment and evaluation of social behaviors on the widely used Pepper platform	Yes	Yes	Yes	Focused on Pepper integration, while SROSP provides a far more extensive architecture
EVA [6]	Open source framework for interactive social robots	Face tracking, speech recognition, dialog management	Provides full pipeline for perception, interaction, and expression	Yes	Yes	Yes	Handles some core social robot capabilities, but SROSP incorporates a much broader range of modules
FURHAT REMOTE API [1]	API for customizing Furhat robot head behaviors	Control of speech, gaze, gestures	Enables integrating Furhat's advanced social expressiveness	Yes	Yes	Yes	Allows custom behaviors through API, SROSP enables fully customizable architecture
MISTY ROBOTICS PLATFORM [5]	Cloud-based development platform for Misty robot	Conversational AI, facial recognition, mapping, navigation	Simplifies programming through web API and Python SDK	Yes	Yes	Yes (potentially)	Cloud platform vs SROSP's on-device modular architecture
MULTI-MODAL PERCEPTION-BASED ASSISTIVE ROBOTIC SYSTEM FOR THE ELDERLY [7]	Elderly assistance robot; integrates vision, speech, touch sensing. Provides real-time behavior interpretation, assistance, personalized interactions and adaptive behavior.	Multi-modal sensing. Real-time perception. Robotic assistance. Personalized interactions.	Enhances quality of life for the elderly through personalized assistance and improved safety.	Yes	Yes	Yes	Overlapping features: multi-modal sensing, real-time perception. Additional features: GUI, control panel, change settings. Aligns with SROSP's goals.
ROBWORK	An open framework for developing robot applications using ROS.	APIs for robotics tasks such as kinematics, dynamics, planning, and control.	General robotics framework, not specific to social robots	Yes	Yes	No (industrial-oriented)	SROSP - Integrated architecture with advanced sensors and processing. RobWork - Modular modeling and simulation without higher-level perception/interface support.
FETCH ROBOTICS SOFTWARE	Open-source software for Fetch mobile manipulators [3]	Navigation, mapping, manipulation planning, perception	Provides a high-level application programming interface (API) allowing the development of task-oriented applications, not specific to social robots	Yes	Yes	Yes (potentially)	SROSP - Modular/distributed design with advanced processing/sensors. Fetch Robotics Software - Monolithic controllers with basic interfaces
PEBBLES ROBOTS	Educational AI tools developed by Anthropic to teach children programming basics	Simple designs like movable blocks could be programmed visually. Safety and ethics were emphasized in their design and use	The PEBBLES project explored how to design AI systems from the ground up with safety, education, and human values as top priorities	Yes. Partially	Yes	Yes	SROSP - Distributed control with diverse sensors and interfaces. PEBBLES - Social interaction through the fixed-mobile platform; less modular customization.
PR2	Personal Robot 2	Mobility assistance, exercises	Contributions to robotics research by demonstrating the capabilities of ROS through its mobile manipulation platform	Yes	Yes	Yes (potentially)	Both have modular node-based software design SROSP adds multi-modal control, custom interfaces, distributed processing across edge/cloud, and broader integration.

TABLE 1: COMPARISON OF PLATFORMS FOR DEVELOPING SOCIAL ROBOTS WITH THE PROPOSED PLATFORM

Platform	Description	Key Features	Contribution	Open Source	ROS-based	Specific to social robots	Comparison to SROSP
HARMONI	An Open-source Tool For humans and Robot Modular OpeN Interaction	Supports complex behavior composition Leverages state-of-the-art external libraries like MediaPipe. Behavior trees	Enables rapid prototyping of social robot interactions	Yes	Yes	Yes	

holistic, flexible HRI systems beyond current solutions limited in their customizability and scope.

Proprietary frameworks like NAOqi [12] and REEM [13] provide integrated toolchains for specific robot models. However, unfortunately, these frameworks cannot be easily extended or modified.

The AMIRO framework [2] enables social robot behavior development and deployment focused on dialog, emotion recognition, and navigation on the Pepper robot. However, it is tailored specifically for Pepper and has limited customizability.

Cloud-based platforms like Misty Robotics [6] facilitate development through APIs but rely on remote rather than on device processing. The Furhat Remote API [1] also provides an interface for customizing behaviors on the Furhat robot head but does not offer full control.

Open source options like Opsoro [14] and FLEX-SDK [15] enable accessible prototyping of behaviors but incorporate a narrower range of capabilities compared to SROSP. The EVA framework [7] handles core social robot capabilities, but not the comprehensive range proposed here.

ROS-based frameworks like Polonius [16] provide building blocks for social robots but lack the complete architecture proposed here. Robotics platforms like Tiago [17] focus more on service applications rather than social intelligence. Assistive robots for the elderly [8] have incorporated multi-modal perception and HRI, but have not focused on rapid prototyping workflows. In contrast, SROSP delivers an on-device modular architecture unmatched for quickly developing social robot behaviors.

In contrast, SROSP delivers an on-device modular architecture unmatched in breadth and customizability for social robotics research. The ROS-based design promotes reusable and flexible integration of capabilities like perception, expression, and interaction. This represents a significant advance, enabling robotics researchers to thoroughly explore social HRI possibilities [18].

The proposed framework aligns with interaction design best practices [9] [10] through its adaptability to different embodiments and behaviors. In summary, SROSP fills a need for rapid prototyping of customized social robot systems with a comprehensive range of modules.

II. SYSTEM ARCHITECTURE

A. Overview

We present a novel modular and customizable software architecture for developing social robot capabilities using ROS. The proposed design consists of key layers to enable the integration of advanced perceptual and reasoning skills required for natural HRI.

A distributed graphical user interface layer allows for personalized remote access to various devices. A centralized controller promotes contextual robot behaviors through asynchronous coordination of independent components. Computational pipelines are partitioned across ROS nodes to optimize processing.

Core robot functions like multimodal sensing, activity recognition, and emotional expression are abstracted as interoperable services. This standardized interface approach streamlines the rapid mixing of capabilities.

The architecture advocates distributed pipelines for synchronized real-time processing of streaming data. Modularity allows customized interface adaptations and behavioral modifications.

This flexible framework is uniquely suited to provisioning next-generation robot assistants, companions, and tutors with the high-level perceptual and socio-emotional sophistication needed for meaningful social engagement with humans.

The proposed architecture aims to advance the state of the art by empowering researchers to quickly prototype and rigorously evaluate novel applications of human-robot interaction. We believe its synergistic design choices can significantly accelerate progress in the field.

B. System Design

The SROSP architecture incorporates modular hardware and software components tailored for social robot capabilities. The overall system architecture consists of four main layers as shown in Fig. 4:

1. Graphical User Interface (GUI) Layer: Allows remote monitoring and control of the robot via a web interface.
2. Web Layer: Manages user requests, parses the GUI, coordinates services between GUI and other layers, and handles the database
3. ROS Layer: Handles low-level robot I/O, sensor processing, and actuator control, while providing reusable perception, analysis, and generation capabilities.

4. Hardware Layer: Handles the robot's hardware (Camera, mic, actuators, etc.)

Graphical User Interface Layer: The GUI is a web-based control panel developed using HTML, CSS, and JavaScript. This control panel provides remote access to monitor and control the robot. It allows users to define temporary sequences of multimedia behaviors combining specific face and sound pairings. A live video feed from the robot's camera can be viewed with options to turn facial landmark analysis graphics overlay on or off. Virtual joystick controls enable teleoperating the robot's movements in real-time. Future work may include additional features to further enhance the remote control and visualization capabilities.

Web Layer: The web layer itself consists of four different layers: 1-Database 2-Django Web application 3-Python WSGI 4-Web Server

The Django web application acts as the central router and coordinator. It exposes RESTful APIs to integrate various services. Core functions like broadcasting behaviors and caching sensor states are implemented as reusable app modules.

The controller broadcasts the facial expression to a specific URL. The broadcast consists of a JSON payload containing the URL of a video file depicting the expression's face and optionally the URL of an audio file containing a corresponding voice/speech sample. This enables remote systems to represent the visualized and vocalized rendering of the detected expression. The co-located media assets potentially enrich expression perception. From hereon we call this JSON an Exp message. Each Exp sent from the user is stored in the database and then triggers the client(e.g. the Android device acting as the robot's face) to request an update from the server and receive the latest set expression. The client's web server continues to stream the media on a loop until a new trigger arises. This methodology saves bandwidth and system overload by preventing frequent requests of the Android client for real-time updates.

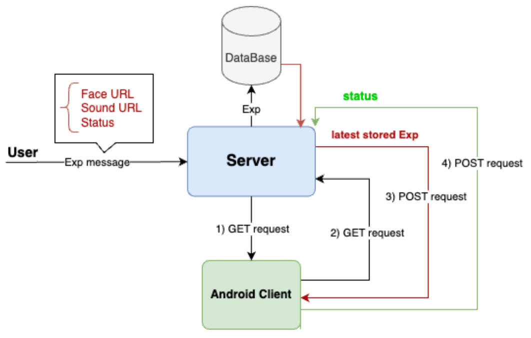

Figure 1: The Exp request handling and updating behavior pipeline (The requests are numbered in the sequence of occurrence.)

The currently used Python WSGI is Gunicorn, followed by the nginx web server for handling processes like SSL/TLS termination, caching, load balancing, and automatically restarting Gunicorn worker processes if needed. By combining Gunicorn and Nginx, the web layer of a robotic platform becomes very robust and customizable. Gunicorn handles rapidly serving application code to power the robot's capabilities. Nginx sits in front, utilizing its advanced networking and process management to securely route real-time HTTP requests to Gunicorn from user interfaces. This allows user-facing elements to be accessed flexibly via local or remote connections with high performance. Features like automatic process restarting in Nginx enhance reliability under varying workloads. Together, Gunicorn and Nginx provide the core web infrastructure required to build and operate sophisticated robotics systems, delivering dependable service for mission-critical robotic applications and empowering the development of optimized user experiences.

3) ROS Layer: The ROS layer executables follow the following structure pattern:

- scripts

- basics

- motion

- Controller node (Actuator)

I/O capturing, and basic processing nodes, such as landmark or emotion detection

- appliances

- middleware

[data processing routers]

- user interface (Django web application)

- services

[I/O data processing, such as gaze detector, speech feature extraction, etc.]

- tools (other modules, datasets, or libraries used) - tests

The system distributes computationally intensive tasks like computer vision and audio processing across separate ROS nodes, improving parallel efficiency. Nodes in the "basic" directory handle robot I/O inputs and initial processed data, that need to be published and accessed constantly for research purposes. The nodes in "middleware" manage to route heavier data processing pipelines. These nodes call the corresponding services in the "services" directory and publish the output to the defined topic. This design facilitates customization in two ways. First, users can redefine individual nodes to update algorithms or I/O. Second, launch file configuration determines node inclusion/exclusion in experiments.

Fig. 2 illustrates the proposed ROS-based computational graph for real-time multi-task computer vision processing. At the start of the pipeline, raw color images and the camera information are published by the /video_stream node from an RGB camera. The images and the camera information are published to the /image_raw and /camera_info topics respectively for downstream consumption.

The core image processing is handled by the /opencv_client node, which bridges ROS and OpenCV. It transforms subscribed image and calibration data into an OpenCV image format before publishing preprocessed images to the /image_cv2 topic. Here we must emphasize that since ROS does not support multidimensional array messages, openCV data such as the frame and landmarks are first transformed from Numpy arrays to lists of lists, and then published as custom messages. Other system limitations are also overcome following this methodology.

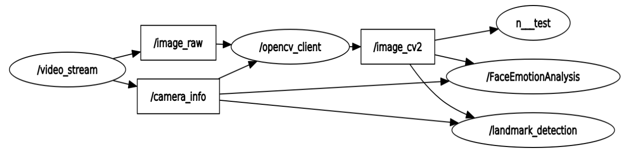

Figure 2: Image processing pipelines for synchronized real-time processing of streaming data

Two computer vision pipelines branch off at this point. The /FaceEmotionAnalysis node implements the DeepFace m8odule [20] to perform facial emotion classification. In parallel, the /Landmark_detection node utilizes MediaPipe [19] for facial landmark localization. Detected landmarks then get utilized in other sensory data processing services such as gaze_detector service as shown in Fig. 3.

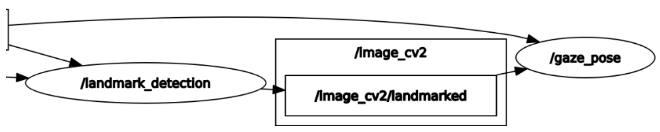

Figure 3: Utilizing the detected landmarks for gaze detection service

This modular architecture supports various extensions and applications. For example, face recognition can be added by calling relevant services or subscribing to the /im age_raw/landmarked topic to bypass early processing steps. Our goal is to provide a configurable toolkit that equips researchers with fundamental algorithms and interfaces through ROS, rather than an all-in-one solution. This is intended to streamline the prototyping of social robotics concepts by furnishing the necessary building blocks to pace development. The extensible design aims to facilitate expedited workflow in collaborative robotics research.

The remaining capabilities of the proposed system follow the same data processing protocol.

Proposed ros-based functionalities include broadcasting camera image converted to opencv image (as custom-defined messages for lists of lists since ros doesn't yet support 3D arrays), broadcasting landmarked camera input image, broadcasting audio input as audio_common messages, Random Forest Classification for audio sentiment analysis, HAZM library for transcribed speech word tokenizing, ParselMouth Praat speech feature extraction, Google speech recognition API for speech-to-text, Microsoft Edge tts for text-to-speech, Google Mediapipe for full body landmark detection and then for gaze estimation, and DeepFace for facial expression estimation for expression imitation.

Finally, the test units for the whole system can be found in the tests directory. Each unit is written for a specific node using standard unittest and rosunit libraries.

III. CONCLUSION

In summary, SROSP introduces an innovative modular architecture for social robotics development that significantly advances the state-of-the-art. Key differentiators include comprehensive high-level perception and expression capabilities packaged as reusable services. A standardized interface approach streamlines integrating capabilities. Results from upstream tasks trigger downstream processing without delay, ensuring minimum latency between stages. While tight coupling exists between chained services, this is outweighed by performance gains. Parallel asynchronous execution amid coordination yields near real-time insights. Reusability is also preserved since services maintain independence. Distributed computational pipelines optimize multi-modal processing. Validation demonstrated lowering barriers to porting behaviors across systems. Overall, SROSP empowers researchers to thoroughly explore HRI through expedited prototyping of innovative applications.

- Provide more detailed evaluation metrics to quantitatively demonstrate SROSP's impact on development timelines. For example, report exact time savings percentages or numbers of lines of code reduced.
- Include validation results comparing SROSP performance (e.g. latency, throughput) to other common robotics frameworks to substantiate claims of speed/scalability advantages.
- Expand theoretical discussion of how SROSP aligns with best practices for modular software design, distributed computing architectures, reusable components, etc. This helps position it as an advancement from an engineering perspective.
- Describe the system/hardware configuration used for testing to ensure results are replicable. Details like CPU/GPU specs, network setup, and software dependencies are important.
- Discuss any limitations, challenges, or areas for future improvement. All systems have room to grow, and acknowledging this lends credibility.

References

1. "Start," Furhat Developer Docs. https://docs.furhat.io (accessed Aug. 02, 2023).
2. A. S,tefania Ghit,aˇ et al., "The AMIRO Social Robotics Framework: Deployment and Evaluation on the Pep per Robot," Sensors, vol. 20, no. 24, p. 7271, Dec. 2020, doi: 10.3390/s20247271.
3. A. O. Elfaki et al., "Revolutionizing Social Robotics: A Cloud-Based Framework for Enhancing the Intelligence and Autonomy of Social Robots," Robotics, vol. 12, no. 2, p. 48, Mar. 2023, doi: 10.3390/robotics12020048.
4. A. Tanevska, F. Rea, G. Sandini, L. Cañamero, and A. Sciutti, "A Socially Adaptable Framework for Human-Robot Interaction," Frontiers in Robotics and AI, vol. 7, Oct. 2020, doi: 10.3389/frobt.2020.00121.
5. "Home page," Misty Robotics, May 02, 2022. https://www.mistyrobotics.com/ (accessed Aug. 02, 2023). [6] "EVA - An open-source social robotics platform." https://eva-social-robot.github.io/#features (accessed Aug. 02, 2023).
6. C. Mollaret et al., "A multi-modal perception based assistive robotic system for the elderly," Computer Vision and Image Understanding, vol. 149, pp. 78–97, Aug. 2016, doi: 10.1016/j.cviu.2016.03.003.
7. A. Sciutti, M. Mara, V. Tagliasco, and G. San dini, "Humanizing Human-Robot Interaction: On the Importance of Mutual Understanding," IEEE Technology and
8. Society Magazine, vol. 37, no. 1, pp. 22–29, Mar. 2018, doi: 10.1109/mts.2018.2795095.
9. T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," Robotics and Autonomous Systems, vol. 42, no. 3–4, pp. 143–166, Mar. 2003, doi: 10.1016/s0921-8890(02)00372-x.
10. P. Alves-Oliveira, P. Arriaga, A. Paiva, and G. Hoffman, "YOLO, a Robot for Creativity," in Proceedings of the 2017 Conference on Interaction Design and Children, Jun. 2017. Accessed: Aug. 03, 2023. [Online]. Available: http://dx.doi.org/10.1145/3078072.3084304

11. D. Gouaillier et al., “Mechatronic design of NAO humanoid," in 2009 IEEE International Conference on Robotics and Automation, May 2009. Accessed: Aug. 03, 2023. [Online]. Available: http://dx.doi.org/10.1109/robot.2009.5152516

12. F. Ferro and L. Marchionni, “REEM: A Humanoid Service Robot," in ROBOT2013: First Iberian Robotics Conference, Cham: Springer International Publishing, 2014, pp. 521–525. Accessed: Aug. 03, 2023. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-03413-3_38

13. C. Vandevelde and J. Saldien, “Demonstration of OPSORO - an open platform for social robots," in 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Mar. 2016. Accessed: Aug. 03, 2023. [Online]. Available: http://dx.doi.org/10.1109/hri.2016.7451853

14. P. Alves-Oliveira, K. Mihata, R. Karim, E. A. Bjorling, and M. Cakmak, “FLEX-SDK: An Open Source Software Development Kit for Creating Social Robots," in Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology, Oct. 2022. Accessed: Aug. 03, 2023. [Online]. Available: http://dx.doi.org/10.1145/3526113.3545707

15. F. Rietz, A. Sutherland, S. Bensch, S. Wermter, and T. Hellström, “WoZ4U: An Open-Source Wizard-of Oz Interface for Easy, Efficient and Robust HRI Experiments," Frontiers in Robotics and AI, vol. 8, Jul. 2021, doi: 10.3389/frobt.2021.668057.

16. J. Pagès, L. Marchionni, and F. Ferro, “[PDF] TIAGo: the modular robot that adapts to different research needs".

17. L. Manso, P. Bachiller, P. Bustos, P. Núñez, R. Cintas, and L. Calderita, “RoboComp: A Tool-Based Robotics Framework," in Simulation, Modeling, and Programming for Autonomous Robots, Berlin, Heidelberg: Springer Berlin Heidel berg, 2010, pp. 251–262. Accessed: Aug. 03, 2023. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17319-6_25

18. K. Dautenhahn, “Socially intelligent robots: dimensions of human–robot interaction," Philosophical Transactions of the Royal Society B: Biological Sciences, vol. 362, no. 1480, pp. 679–704, Feb. 2007, doi: 10.1098/rstb.2006.2004.

19. C. Lugaresi et al., “MediaPipe: A framework for building perception pipelines," arXiv.org, Jun. 14, 2019. https://arxiv.org/abs/1906.08172

20. Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the gap to human-level performance in face verification," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 2014. Accessed: Aug. 30, 2023. [Online]. Available: http://dx.doi.org/10.1109/cvpr.2014.220