RoBorregos Team Description Paper for RoboCup@Home OPL 2025

Adán Flores Emiliano Flores Iván Romero Alexis Chapa
Francisco Salas Alejandra Coeto David Vázquez
Diego Hernández Oscar Arreola Yair Reyes
Alejandro Guerrero José Domínguez Gerardo Fregoso
Gilberto Malagamba Alejandro González Jocelyn Velarde
Leonardo Llanas Marina Villanueva Angel Cervantes
Alberto Muñoz Carlos Vázquez

School of Engineering and Sciences, Tecnológico de Monterrey
C.P. 64849, Monterrey, N.L., México
https://athome.roborregos.com/
https://github.com/RoBorregos/home

Abstract. This paper presents the status of the service robot FRIDA, developed by team RoBorregos from Tec de Monterrey. The platform consists of a differential base and a 6-DOF robotic arm, integrated with custom hardware. The current work focuses on research and development of several hardware and software modules. These modules are encapsulated with a task planner to manage states, integrating HRI(Human-to-Robot Interaction), perception, and movement modules. The current year of research and development has been focused on implementing a stable automatic object detection training pipeline; the use of local LLM's for command interpretation and task planning; the use of a local VLLM for person's attributes; person recognition and re-identification; map segmentation for custom environment contextualization; development of a custom manipulation planner using cartesian and joint controllers for constrained manipulation scenarios; Overall, the work developed toward each area has contributed significantly to several research and development projects at ITESM.

1 Introduction

Established in 2015, RoBorregos is ITESM's representative robotics team, composed entirely of undergraduate students from diverse academic disciplines. Since

its foundation, the team's focus has been to participate in autonomous robotics competitions. At RoBorregos, @Home has been the largest project, whose development started 4 years ago. Over this time, the team has had several participations in the @Home league. Participations started in the Mexico's @Home Beginners in 2021 and 2022, earning recognition for the best HRI and Object recognition systems; In 2021 team members obtained the "most consistent solution" in the AirLab Stacking Challenge. In 2022 the participation at the IROS @Home Simulation achieved 5th place. In 2023, our first participation in the OPL took place at Mexico's @Home competition. This year, the team achieved second place in the @Home Mexico competition and its first participation in the OPL at RoboCup. Furthermore, the team has an active collaboration with ITESM's School of Engineering, building software and hardware developments to apply them in academic courses and industrial context. Three papers have been published for academic conferences such as EDUNINE, EDUCON and Conference on Learning Factories describing the impact of these technologies in the learning process.

This Team Description Paper provides an overview of our research and development to achieve a robust implementation to perform the tasks proposed by the RoboCup@Home competition.

2 Research Focus and interests

The team consists of 19 undergraduate students (B.Sc.) working both on robot development and the associated research. The team is advised by two professors who provide technical and research guidance. RoBorregos' primary platform is a mobile base (EAI Dashgo B1) paired with a 6-DOF robotic arm (xArm6). The main research focus is integrating multiple systems into a cohesive and functional architecture. Additionally, RoBorregos focuses on promoting robotics within the community by attending local events, open sourcing all the software, and providing mentorship to the community. This year, significant progress was achieved in several areas of interest, such as:

1. Migrating and benchmarking several models (STT, TTS, LLMs, embeddings) from remote to local alternatives.
2. Dividing complex tasks into smaller sub-modules for better encapsulation.
3. Transforming the infrastructure through Docker-based containerization.
4. Implementing person-tracking and re-identification capabilities.
5. Exploring newer, more accurate AI models to keep pace with rapid advancements on the field.

3 Robot architecture

Our robot's software architecture was developed in ROS (Robot Operating System) using the Noetic distribution. It is a behavior-based architecture centered in the Task planner node. This node serves as the decision-making layer in the

process, processing voice commands and delegating specific tasks to the appropriate sub-modules. The Task planner is responsible for analyzing every task and executing them with the sub-modules integrations. Each sub-module represents fine-grained abilities for perception and actuation.

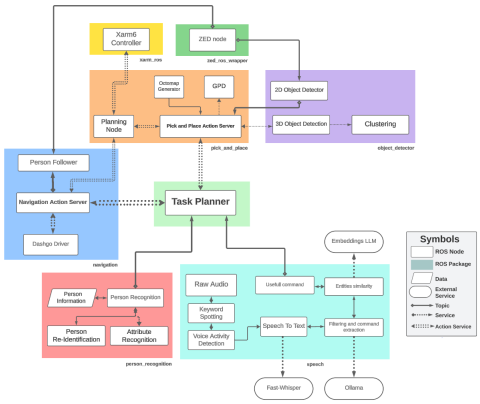

Fig. 1: Diagram of generalized interactions between software modules

4 Scientific contributions and current work

This section delves into further detail about the highlights of development and achievements per area.

4.1 Integration

Part of the efforts during this season was to transition from monolithic systems to horizontal scaling through the virtualization of each module. The use of containerization technologies, mainly Docker and NVIDIA-containers, enabled us to distribute the computational load and enhance the possibilities of using the developed nodes for other applications. The transition performed this year enhances development by standardizing the environment and allows more developers to build on top of the current infrastructure. Furthermore, this work helps us maintain granular control of the compute workload over the distributed system.

Task Planning. The task planner is based on a task manager that enables the interaction of all the modules. It works by defining a dynamic queue of tasks and executing them in a finite state machine, the queue is either invoked via EGPSR commands, or a preset queue of states like other tasks. Each capability is

managed by a sub-state machine for each of the robot's capabilities. Furthermore, the task planning node enables a dynamic context for the environment and certain events that will enhance the context for commands. Part of this is mainly the command extraction from HRI by enhancing it with the possible actions and interactions allowing this to expand the fallbacks for failing states and allow restructure in the queue of tasks if needed.

4.2 Object Manipulation

The developed pipeline enables interaction with a 6-degree-of-freedom (DOF) robotic arm and its gripper for object manipulation tasks by incorporating image processing, perception, and motion planning components. The manipulation module consists of several components to achieve the required actions, including the object picking, placing phases and the motion planning.

The **object picking** phase consists in extrapolating the 2D object detection data with the 3D point cloud. The use of PCL [1] allows the algorithm to process the resulting perception data to extract planes thought RANSAC [2], and segment corresponding clusters to the detected objects. We also make use of the GPD [3] library to process the detections and use MoveIt! [4] to create collision-free trajectories to the grasps detections. In the **planning phase** primitive movements were used for pick/place for constrained planning based on the arm's move joints. This allowed for faster planning while also ensuring the end effector's orientation to stay the same in the trajectory. This approach is mainly used for the serving breakfast task, where this approach doesn't require collision avoidance. Otherwise, this approach requires a collision avoidance alternative. Lastly, the **placing** algorithm also relies on isolating the table surface with RANSAC [2], the resulting surface is then used to create a heat-map based on space of availability. Creating a Gaussian-distribution heat on available space and taking the region with the maximum accessible space and using its center as the placing pose [Fig. 3].

4.3 Computer Vision

Face recognition. To recognize people with high confidence, facial features were selected as the best approach, focusing on face landmarks and distances. Therefore, DLib's face-recognition model [5] was used. In general terms, the recognition node loads previous detections before processing each frame. For this, new encodings are extracted from each face detected and compared to saved encodings in order to identify if it is a known person. Nonetheless, comparing each face to saved faces on each frame is exponentially complex, so a tracking feature was implemented by comparing each face location to its previous position, thus inferring that the face belongs to the same person in the previous frame if it lays within a specified threshold.

Person re-identification. Even though face recognition offers high confidence for re-identifying individuals, it has limitations when people are too distant or

not visible. Hence, an alternative tracking method is needed to handle scenarios like following a person who exits and re-enters a robot's field of view.

The proposed solution consists of three phases: detection, tracking, and re-identification. Detection and tracking were handled using the YOLOv8 model and ByteTracker [6] algorithm, which assigns IDs to visible individuals across frames. Nonetheless, this implementation cannot re-identify a person if they go behind large objects or leave the frame, which introduced the necessity of the third phase. Thus, based on the paper Multi-Camera People Tracking [7], a feature extractor was implemented. For this, the repository Person reID baseline PyTorch [8] was used as a base to train different models using the Market1501 dataset. More specifically, the models DenseNet, ResNet50, NetSwin, NetSwinV2 and OsnetX1 were trained and tested using Rank-K, metric that measures the accuracy of retrieving the correct result within the top-K ranked predictions [Table 1]. Even though The NetSwin and OsnetX1 provided the best results, we opted for the DenseNet model as it was the lightest and performed accurately on query tests [Fig. 5, 6]. Nonetheless, the NetSwin model would be recommended for computers with more computational power.

The trained model generates a feature vector for each person's image, enabling identification of previously detected individuals using cosine similarity. Finally, To improve accuracy, only individuals with visible key body landmarks, identified through MediaPipe [9], are analyzed.

Pose, action and attributes. A pose and action detection module was implemented using body landmarks from the MediaPipe model. Joint angles were calculated to estimate body positions, enabling the detection of poses or actions, such as raising hands based on hand placement and shoulder angles.

For visual model descriptions the locally hosted VLLM Moondream [10] was used model, which is a prompt power visual analysis model. Given a contextualized frame with a specific prompt, the model provides an accurate description of the image enhanced with the prompt analysis and response.

Object detection. A pipeline was developed that produces a dataset with the necessary encoding to be trained with YoloV8. This pipeline utilizes two technologies. The first is GroundingDINO [11], which enables us to detect objects of our interest and obtain their bounding boxes. The second is SAM (Segment Anything Model) [12], which we use to crop images from their original backgrounds and create images containing only the desired objects. To achieve this, SAM requires the bounding boxes of the objects we want to segment, which are provided by GroundingDINO. Additionally, a module was developed that transforms any object detection dataset into an object segmentation dataset. This is achieved using SAM, leveraging the YOLO bounding box annotations and re-annotating the labels with a segmentation notation.

4.4 Human-Robot Interaction (HRI)

Pipeline Overview. The development of HRI consists of 2 pipelines: speech and NLP processing. The speech pipeline has several modules to achieve robustness while being computationally efficient (i.e. ensure resource-intensive models only infer voiced audio). The first module of speech is the Keyword Spotting node, which was implemented using porcupine [13]. After the keyword is detected, audio is recorded until voice is no longer detected using Silero VAD [14]. The audio captured is then inferred using Whisper. Finally, the NLP pipeline processes the interpreted text by parsing the string into commands using an LLM. As a strategy to deal with malformed LLM outputs, the resulting commands are matched to the closest valid action according to the robot context and capabilities, found via cosine similarity by embedding the actions if there is no exact match. [Fig. 7]

Filtering and Command extraction. For several tasks such as General Purpose Service Robot (GPSR), it is crucial to understand user instructions and divide them into commands that the robot can perform. This was previously achieved using a fine-tuned version of OpenAI's 3.5 turbo model. However, this posed some challenges, specifically with the latency of the API and the need to have internet to access the model. To address these problems, a local implementation was made using Llama 3.2 3B. By fine-tuning the model with a custom dataset, the new implementation yielded comparable results while reducing latency. It is worth noting that the results of the fine-tuning process were published to HuggingFace.

Entities Similarity. After the command interpreter has given the output, the next layer is the Entities Similarity Comparison. This layer allows the robot to ground the key components of the command (action, location, item, and item category). This ensures that parsed commands are matched to the closest components in the robot's knowledge base.

This layer is implemented by leveraging embedding models, which allow extracting the semantic value of a sentence into a vectorial output and perform comparison using cosine similarity. The main contribution consists of the benchmarking of multiple popular embedding models to select the best fit for specific use cases. The OpenAI `text-embedding-3-small` and `text-embedding-3-large` [15] models were also benchmarked to have a reference point.

The benchmarking consists of having a predefined set of actions, locations and items, each with a set of synonyms that were generated using GPT-4. After that, models are evaluated using a variety of metrics. [Tables 2, 3]

Changes in STT Model. For the Speech to text node, we use Whisper [16]. A benchmark comparing it with Faster-Whisper models was made with audios recorded in different noise scenarios which yielded half the translating time and

higher accuracy, concluding in the migration to this model. Additionally, "hot words" were integrated and are dynamically changed depending on the context of the task the robot is performing. This makes translation more robust by improving accuracy for non-common words in case there is a resemblance.

4.5 Navigation

A navigation and mapping system was implemented to enhance FRIDA's functionality. It incorporates simultaneous localization and mapping (SLAM), dynamic goal management, map contextualization, and safe approach strategies, ensuring improved accuracy, modularity, and adaptability in diverse operational environments. The SLAM approach was implemented using the ROS Gmapping [17] package for mapping and the AMCL [18] algorithm for localization. Navigation is managed by the ROS Navigation Stack [19]. Goal transmission is simplified through an action server that uses predefined poses (e.g., Kitchen, Room), defined by spatial coordinates (x, y, z) and orientation quaternions. This modular design facilitates integration of goals into the autonomous system.

To enhance real-time awareness, the map is segmented into coordinate-based sections linked to contextual data, allowing the robot to determine its position at any moment. A safe table approach algorithm was developed to optimize the robot's positioning for manipulation tasks. Additionally, person-following performance is improved by adjusting the Xarm's rotational velocity based on the target bounding box's position relative to the camera frame's center. A proportional error is calculated when deviations occur and is fed into the rotational velocity controller. This strategy is refined by considering the base's rotational direction, ensuring smooth and consistent tracking.

5 Conclusions and future work

The team has developed a robust and functional set of general-purpose modules, integrated in a layered behavior-based architecture. These features enable FRIDA to perform dynamic tasks interpreted and solved with its skills.

Our participation in RoboCup 2024, allowed us to identify areas of opportunity and inspired us to keep researching to improve each module, enhancing areas like Manipulation with newly introduced hardware accelerated solutions for planning, Navigation with the planned transition to ROS2, HRI by making full-use of current LLM and generative models locally and Computer Vision by improving the tracking pipeline. Lastly, research and development began for a holonomic base, aiming to include and extend all functionality from the Dashgo B1, expected to be ready for use before 2026.

References

1. Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011.

2. José María Martínez-Otzeta, Itsaso Rodríguez-Moreno, Iñigo Mendialdua, and Basilio Sierra. Ransac for robotic applications: A survey. *Sensors*, 23(1):327, 2022.
3. Andreas Ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14):1455–1473, 2017.
4. Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit![ros topics]. *IEEE robotics & automation magazine*, 19(1):18–19, 2012.
5. PyPI. face-recognition 1.3.0, https://pypi.org/project/face-recognition/, 2020.
6. Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *European conference on computer vision*, pages 1–21. Springer, 2022.
7. Zhongyu Jiang Pyong-Kun Kim Kyoungoh Lee Kwangju Kim Samartha Ramkumar Chaitanya Mullapudi In-Su Jang Chung-I Huang Jenq-Neng Hwang Hsiang-Wei Huang, Cheng-Yen Yang. Enhancing multi-camera people tracking with anchor-guided clustering and spatio-temporal consistency id re-assignment. *arxiv*, 2023.
8. Zhedong Zheng, Xiaodong Yang, Zhiding Yu, Liang Zheng, Yi Yang, and Jan Kautz. Joint discriminative and generative learning for person re-identification. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2138–2147, 2019.
9. Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019.
10. Moondream,https://moondream.ai/, 2024.
11. Tianhe Ren, Qing Jiang, Shilong Liu, Zhaoyang Zeng, Wenlong Liu, Han Gao, Hongjie Huang, Zhengyu Ma, Xiaoke Jiang, Yihao Chen, et al. Grounding dino 1.5: Advance the" edge" of open-set object detection. *arXiv preprint arXiv:2405.10300*, 2024.
12. Meta AI. Segment anything model (sam), https://segment-anything.com/, 2023.
13. PICOVOICE. *Porcupine Wake Word, https://picovoice.ai/platform/porcupine/*. 2024.
14. Silero Team. Silero vad: pre-trained enterprise-grade voice activity detector (vad), number detector and language classifier. *Retrieved March*, 31:2023, 2021.
15. OpenAI. *New embedding models and API updates.* OpenAI Platform, https://openai.com/index/new-embedding-models-and-api-updates/, 2024.
16. Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR, 2023.
17. Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007.
18. Wang Xiaoyu, LI Caihong, Song Li, Zhang Ning, and FU Hao. On adaptive monte carlo localization algorithm for the mobile robot based on ros. In *2018 37th Chinese Control Conference (CCC)*, pages 5207–5212. IEEE, 2018.
19. Kaiyu Zheng. Ros navigation tuning guide. *Robot Operating System (ROS) The Complete Reference (Volume 6)*, pages 197–226, 2021.

FRIDA Hardware Description

A modified EAI Dashgo B1 robot was used for mobility purposes and an uFactory xArm 6 was added for manipulation tasks and to mount the depth camera.

Fig. 2: FRIDA

- Base: A chassis with a differential configuration of wheels was used as a base, and motors with encoders were used for odometry purposes.
- Computers: Jetson Xavier AGX for manipulation and vision tasks and a Jetson Nano connected to it via ethernet through a local network that handles the robot's movement and sensor processing.
- Batteries: 24v batteries for both Xarm 6 and Dashgo (including its Jetson Nano), and 14.8v battery for Jetson Xavier.
- Arm: The robot uses an xArm 6.
- Arm gripper: Universal custom gripper, interchangeable with variants such as a parallel gripper, a gripper with higher precision for small objects, and a gripper with higher strength using a stepper motor.
- Vision: Stereolabs ZED2 camera attached to the EOF of the Xarm that runs the vision tasks and transmits data through a USB cable to the Jetson Xavier
- HRI: Seed Studio ReSpeaker Microphone array and a Logitech X100 Wireless Portable Speaker, HDMI LCD touch screen display.
- Navigation: RP Lidar A1.
- Base cover: Custom sheet metal manufactured structure for passive heat dissipation
- Robot dimensions: height: 186 cms (fully extended), width: 42 cms
- Robot weight: 37 kg.

Additionally our robot incorporates the following devices:

- Power indicator for the Xarm 6 and Jetson Xavier batteries
- 5V switch that enables local data transmission through a ethernet network
- Independent emergency stop devices for Jetson Xavier, Xarm controller and Dashgo
- Customized gripper controller

Robot software and hardware specification sheet

Robot's Software Description

For our robot we are using the following software:

- Platform: Robotic Operating System (ROS), Noetic distribution
- Integration: Docker
- Navigation: ROS Navigation Stack
- Person analysis: Torch, DLib face-recognition YOLOv8, Mediapipe, Bytetrack, Moondream
- Speech recognition: Whisper
- Voice activity detection: Silero VAD
- Keyword spotting: Porcupine
- Text processing: OpenAI GPT 3.5, Llama 3.2, sentence-transformers/all-MiniLM-L6-v2
- Object detection: YoloV8, Segment anything, GroundingDino
- Manipulation planning: MoveIt!
- Point cloud proccessing: PCL, RANSAC

Appendices

1 Manipulation

1.1 Placing Gaussian-Distribution Method

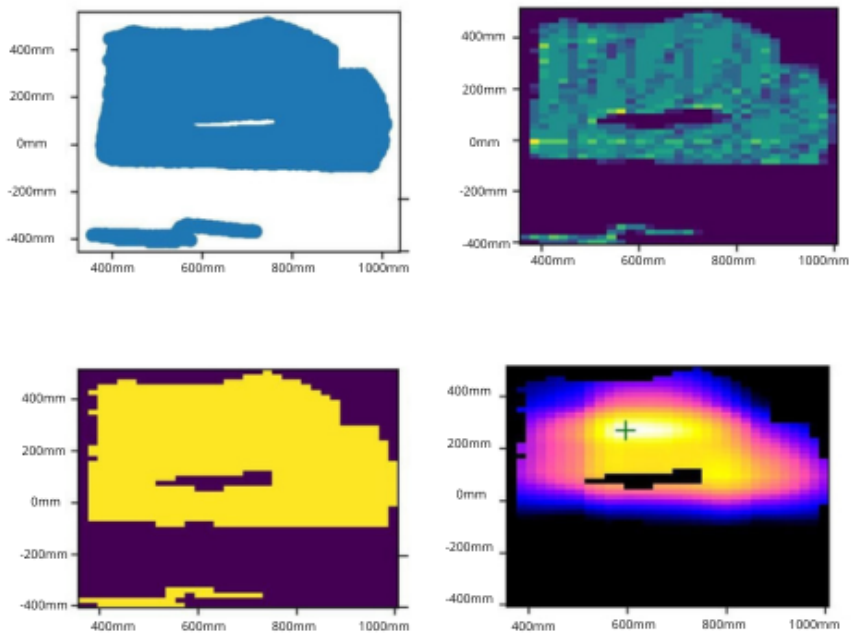

Fig. 3: Illustration of the Gaussian distribution method

2 Computer Vision

2.1 Overview of vision capabilities

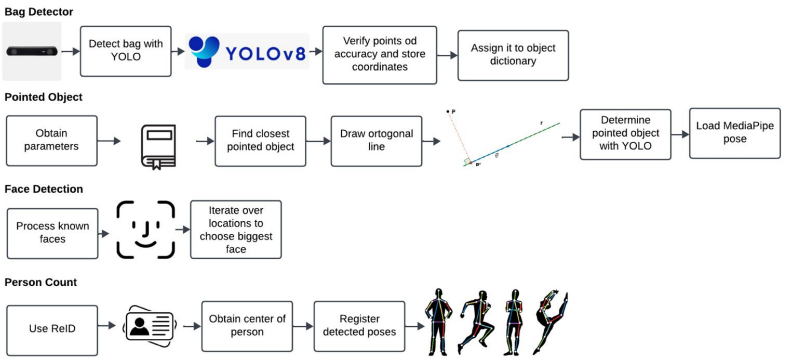

Fig. 4: Overview of computer vision capabilities

2.2 Rank-K Accuracy for Re-Identification models

Model	R1	R5	R10	mAP
ft_net_dense	0.807002	0.905296	0.925943	0.635984
ft_resNet50	0.884204	0.951306	0.968527	0.712407
ft_net_swin	0.923694	0.974466	0.984561	0.793326
ft_net_swinv2	0.915677	0.968527	0.983373	0.779673
osnet_x1_0	0.942	0.979	0.987	0.826

Table 1: Rank-K values for different models

2.3 Re-Identification Query Tests

Fig. 5: Re-Identification models query tests 1

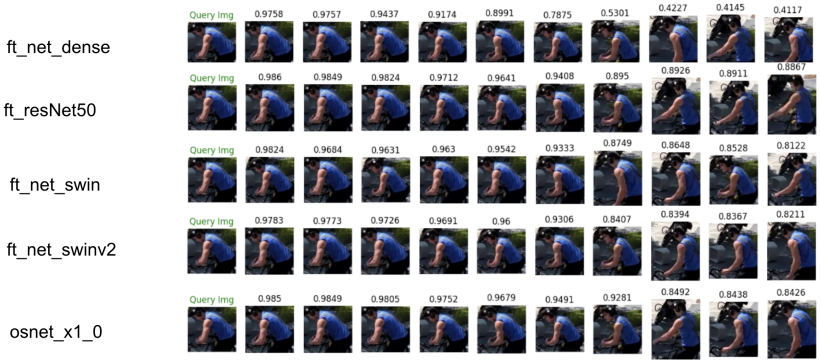

Fig. 6: Re-Identification models query tests 2

3 Human Robot Interaction

3.1 Benchmarking results for entities similarities

Original Word	Synonyms
Approach	Move closer, advance, get nearer
Living_Room	Lounge, Sitting room, Family room
Cereal_Box	cereal_pack,breakfast_cereal,grain_box

Table 2: Sample word-synonym conversion used for benchmarking

MODEL	Accuracy(%)	Average embedding time(s)	Peak GPU Memory (mb)
paraphrase-TinyBERT-L6-v2	78.49	0.009	264.63
all-MiniLM-L6-v2	78.49	0.008	95.40
all-MiniLM-L12-v2	76.88	0.009	135.39
all-mpnet-base-v2	67.74	0.011	428.76
multi-qa-mpnet-base-cos-v1	80.65	0.012	426.85
paraphrase-MiniLM-L6-v2	78.49	0.011	95.40
paraphrase-distilroberta-base-v1	72.04	0.011	322.62
stsb-roberta-large	76.88	0.012	1365.72
roberta-large-nli-stsb-mean-tokens	76.88	0.014	1365.72
paraphrase-albert-small-v2	60.75	0.013	54.08
all-roberta-large-v1	46.77	0.015	1365.72
all-distilroberta-v1	56.45	0.015	323.90
multi-qa-distilbert-cos-v1	77.96	0.014	263.86
paraphrase-albert-small-v2	60.75	0.014	52.95
paraphrase-MiniLM-L3-v2	79.03	0.013	74.72
text-embedding-3-small	73.12	0.331	-
text-embedding-3-large	78.49	0.348	-

Table 3: Results of benchmarking

3.2 General HRI Pipeline Illustration

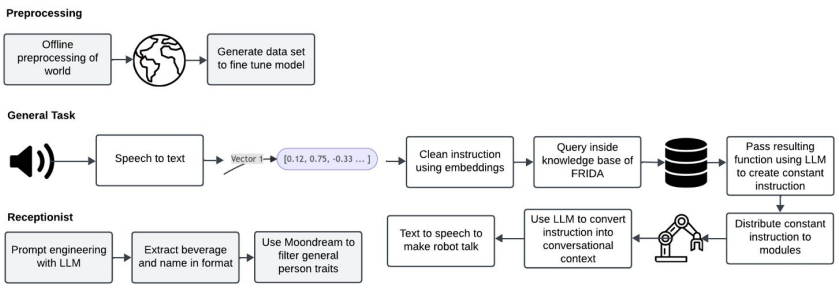

Fig. 7: HRI Pipeline illustration

3.3 Benchmarking STT Models

File (10s)	Size (MB)	Faster-whisper accuracy	Time (s)	Whisper accuracy	Time (s)
test1.wav	1.22	85.7%	0.64	71.4%	1.25
test2.wav	1.22	77.8%	0.71	33.3%	1.44
test3.wav	1.22	71.4%	0.66	57.1%	1.13
test4.wav	1.22	80%	0.70	60%	1.36
test5.wav	1.53	71.4%	4.68	71.4%	4.5
test6.wav	1.83	42.9%	0.63	28.6%	1.03
test7.wav	1.83	90%	0.64	90%	0.87
test8.wav	1.83	83.3%	0.61	66.7%	0.99
test9.wav	1.83	100%	0.62	100%	0.94
test10.wav	1.83	100%	0.58	100%	0.77

Table 4: Sample of Comparison of faster-whisper and whisper implementations