2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

A Framework for Human-Robot Interaction User Studies

Vidyasagar Rajendran[1], Pamela Carreno-Medrano[2], Wesley Fisher[1], Alexander Werner[1], Dana Kulić[1,2]

Abstract— **Human-Robot Interaction (HRI) user studies are challenging to evaluate and compare due to a lack of standardization and the infrastructure required to implement each study. The lack of experimental infrastructure also makes it difficult to systematically evaluate the impact of individual components (e.g., the quality of perception software) on overall system performance. This work proposes a framework to ease the implementation and reproducibility of human-robot interaction user studies. The framework utilizes ROS middleware and is implemented with four modules: perception, decision, action, and metrics. The perception module aggregates sensor data to be used by the decision and action modules. The decision module is the task-level executive and can be designed by the HRI researcher for their specific task. The action module takes subtask requests from the decision module and breaks them down into motion primitives for execution on the robot. The metrics module tracks and generates quantitative metrics for the study. The framework is implemented with modular interfaces to allow for alternate implementations within each module and can be generalized for a variety of tasks and human/robot roles. The framework is illustrated through an example scenario involving a human and a Franka Emika Panda arm collaboratively assembling a toolbox together.**

I. Introduction

With the fast-growing demand for robotic systems capable of interacting with humans in a safe, seamless, and intuitive manner, the development and integration of these systems in interactive and collaborative tasks has seen an extensive research effort in recent years, e.g., [1], [2]. However, while significant progress has been made on advancing the robotic technologies needed to support human-robot interaction (HRI) and human-robot collaboration (HRC) applications (e.g., environment perception and sensing, task planning and decision making, human-aware motion planning), less attention has been given to the development of experimental systems that facilitate the evaluation of these technologies in the context of interactive and collaborative tasks.

In recent years, researchers have endeavoured to address some of the issues with the replicability, standardization, and evaluation of human-robot interaction and robotics research in general. In [3], a model set of tasks for human-robot collaboration applications was proposed. Through this set, the authors aimed at facilitating the experimental evaluation of new algorithms and methods in HRC by reducing the amount of work required to set up and reproduce such experiments. Moreover, the proposed set of tasks also provided a unified framework that standardizes how contributions to the field are compared. Complementary to the standardization of tasks, Murali *et al.* [4] introduced PyRobot, an open-source robotics framework that provides access to datasets, standard algorithm implementations and models that can be used to quickly build a state-of-the-art baseline software system for a robot. In [5], [6], a set of common and standardized metrics to evaluate different aspects of a human-robot team are proposed, including metrics such as performance, interaction effort, and fluency.

However, even with tools such as PyRobot or a standardized set of tasks like the one proposed in [3], the assessment of new methods and algorithms for human-robot interactive and collaborative scenarios continues to be an open challenge. For instance, if a team of researchers proposes a novel human-intention prediction algorithm within the context of a particular HRI application, the researchers will need to validate the algorithm using standalone and prediction-specific tests while also considering the effect of the algorithm on aspects specific to HRI scenarios (e.g., quality of interaction, the performance of the human-robot team, the robustness and autonomy of the robot during the interaction etc. [7]).

Coupled with this challenge in evaluation, HRI scenarios are inherently complex to instantiate since they require a whole component stack beforehand (e.g., low-level motor controllers, path and motion planning, decision making, object detection and identification). This complexity in implementation makes it difficult to compare novel algorithms and methods against existing work in the literature. Even when researchers make their work widely available, most implementations are tweaked to match the needs of the research task and/or context of the application [8]. This makes code reuse for comparison and benchmarking a challenging endeavour.

In order to facilitate the evaluation of new algorithmic and technological contributions within HRI and HRC scenarios, we propose, implement, and illustrate a unifying experimental framework through which HRI researchers can design and conduct experiments. The framework uses ROS (Robot Operating System) middleware [9] and focuses on collaborative tasks with one human and one robot. It also provides a baseline software implementation to aid with the design of user studies and encourage code reuse in a way that will make studies easier to reproduce and compare. Figure 1 depicts the framework at a high-level. The system consists of four main modules: perception, decision, action, and

[1]Vidyasagar Rajendran, Wesley Fisher, Alexander Werner and Dana Kulić are with the Department of Electrical and Computer Engineering, Faculty of Engineering, University of Waterloo, Ontario, Canada {vrajendr, wfisher, awerner, dkulic}@uwaterloo.ca

[2]Dana Kulić and Pamela Carreno-Medrano are with the Faculty of Engineering, Monash University, Melbourne, Australia {Dana.Kulic, Pamela.Carreno}@monash.edu

metrics. All modules have simple and well defined interfaces to enable flexible implementations within each and allow researchers without expertise in one or more of the modules to use the framework. Researchers can also label important events online during the task, and collect objective performance metrics which are calculated automatically within the framework. We evaluate the proposed framework through an illustrative human-robot collaborative assembly task using the Franka Emika Panda arm. The codebase is available under the *GPLv3* open-source license on GitHub[1] allowing for contributions and extensions by other researchers in the field of HRI.

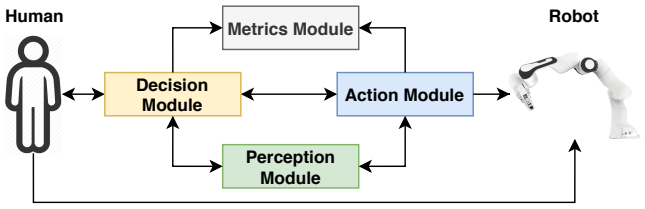

Fig. 1: The high-level block diagram of the proposed HRI collaborative assembly framework.

II. Related Work

Running full HRI studies is a multi-stage process. During the design and conception stage, researchers must identify the collaborative task or interaction context, decide on the robot's autonomy level, including perceptual and action capabilities, and finally design and plan for the different ways in which the robot will interact with the human. During the development stage, researchers are tasked with defining and implementing suitable robot functions and behaviors as well as implementing and/or integrating appropriate interaction modalities (e.g., speech, non-verbal behavior, text). Prior to deployment and validation, researchers must determine appropriate methods of evaluation (e.g., questionnaires, performance-related metrics, physiological signals), plan and decide on the experimental protocol (e.g., number of participants, duration of the study, number of conditions to be evaluated), among other aspects. In this section we provide an overview of systems and tools conceived to facilitate and support HRI and HRC researchers during each of these stages.

Regarding the design stage, Zeylikman *et al.* [3] have recently proposed a reference set of tasks for reuse in HRC experiments. The tasks were designed for a collaborative assembly scenario and can be adapted to investigate many aspects that are relevant during collaboration (i.e, role assignment, task allocation, amount of information available to the human and the robot etc.). In the context of human-robot collaboration in manufacturing, Malik *et al.* [10] proposed a Digital Twin (DT) framework that allows for quick analysis and validation of key aspects such as human-robot task allocation, workstation layout, and human ergonomics. Complementary to this idea, in [11], a ROS based architecture for designing and validating the coordination of tasks that require humans and robots to collaborate was proposed. Using this architecture, human and robot actions defined using off-line programming tools can be executed and tested prior to deployment.

While these frameworks provide a systematic approach to study design they lack the necessary benchmarking and metrics gathering infrastructure needed to objectively evaluate HRI contexts and studies. Metrics are left for the researcher to identify and implement on their own outside of the framework implementations.

At the development stage, development of reusable libraries for the interaction and decision capabilities of a robot has been of great interest. In [12], HRItk, a framework designed to allow for the rapid development and deployment of speech and gesture-based interactive systems within the ROS environment was proposed. Clodic *et al.* [13] proposed a control architecture that allows robots to support the completion of human-robot collaborative tasks and also produce legible and socially acceptable behaviors. To assist with the implementation of a robot's behavior and actions, Paxton *et al.* [14] introduced CoSTAR, a modular and cross-platform system that allows novice end-users to create robust task plans for robots.

Most of the tools and systems reviewed so far are specific to the development of the robot's capabilities and behaviors. It is still up to to the researcher to develop all the other elements that are necessary for the evaluation of these behaviors during interaction. Moreover, most of these tools were designed with a specific context in mind, such as human-robot collaboration in manufacturing and assembly, which limits their application and usability.

At a larger scale, several software frameworks, tools and task sets have been proposed to facilitate research in both robotics and human-robot interaction. The iCub-HRI library, a software framework built to support complex human-robot interaction scenarios on the iCub robotic platform, was presented in [15]. iCub-HRI integrates several components for perception, object manipulation, and social interaction. While a few components of the framework are robot-independent, most are dependent on the iCub hardware and as such, would need to be re-implemented or substituted with alternatives for other robot platforms.

With the aim of allowing non-robotics researchers to focus on building and developing high-level applications and algorithms without worrying about low-level robot controllers, [4] introduced PyRobot. PyRobot is an open-source robotics framework that provides a light-weight, high-level interface on top of ROS and integrates libraries for tasks such as hand-eye calibration, teleoperation, trajectory tracking and SLAM-based navigation. This work provides useful abstractions for robot functionality and is generalizable to many robots but lacks a pathway to full system integration for large scale user studies for HRI.

Fong *et al.* proposed HRI/OS [16], a structured software

[1] https://github.com/hri-group/hri_framework hosts the codebase, documentation, videos and examples.

framework for building human-robot teams. HRI/OS is an agent-based system that supports a variety of user interfaces, enables task-oriented dialogue between humans and robots, and facilitates the integration of different robots through an extensible API. While this work provides a general framework for human-robot interaction, it only tracks robot progress during the task and it is also unclear whether the project is still active and if the existing codebase can integrate well with modern robotic systems.

For metrics tracking in the context of HRI, Hoffman proposed an analytical model for four objective metrics and assessed them in a simulated turn-taking task to move towards a standard benchmarking for human-robot fluency [6]. He surveyed works ranging from 2007 to 2018 that reported these metrics, showing the wide adoption in existing HRI research. While there is usage of a common set of metrics in HRI, a unifying implementation that can automatically generate them is missing. Furthermore, outside of Hoffman's work, other common metrics are often defined in an abstract manner, with their interpretation and implementation left to the researcher. Thus, a fair comparison between systems and studies is currently hard to achieve.

The HRI research community has made consistent efforts towards building tools that enable expert and non-expert users to easily program robots and developing benchmarks for human-robot interaction evaluation. However, a complete experimental framework that can aid HRI researchers in implementing user studies that can be easily compared and reproduced is missing in the field.

III. Design Specifications

The main objective of our framework is to enable researchers to easily and systematically validate HRI components and systems in a consistent and reproducible way. This encourages researchers to take into consideration all the different module implementations required for their study along with which objective metrics they want to report at the end of the study.

A current challenge in HRI is that algorithms for perception, task planning, control and other domains are often developed in isolation without being tested systematically in the context of a full HRI scenario. That is why on a component level, our goal was to provide a framework that is modular and allows for new algorithmic contributions to be tested and evaluated in the context of a full HRI setup. For example, when developing new perception modules, it is common to test the algorithms against state of the art image datasets. While this provides information on the technical capabilities of the algorithm, for use in HRI, it is essential to quantify how the algorithm affects human-robot fluency or other objective (or subjective) metrics. This type of systematic testing of algorithms is currently lacking in the field of HRI.

To address these challenges and fulfill our framework objectives, we followed the approach in [14] and [15], and formulated design specifications to guide the development of the framework:

1) **Modularity and flexibility:** the framework should have simple and well defined interfaces to enable flexible implementations of each module and allow researchers without expertise in one or more of the modules to use the framework.
2) **Interoperability:** the framework should enable interoperability of implementations of each of the modules as long as the basic interfaces between them are satisfied.
3) **Usability:** the framework should be easy to use by researchers working on applications in which a human and robot must interact with each other to accomplish a shared-goal (e.g., collaborative assembly).
4) **Extendability:** the framework should be easily adaptable to a variety of tasks and human/robot roles.

One of our primary goals while developing this framework was to allow for implementations of varying complexity within each of the core modules shown in Figure 1. HRI study implementations vary based on available robotic hardware, experimental setup and participant demographics. To allow for modularity and flexibility, instead of developing a set of sophisticated modules, we defined interfaces between each of the modules while also providing minimal example implementations to aid with the development of new studies.

Interoperability ties in with flexibility in that we want implementations of modules to be easily exchangeable to allow for variants of each module to be tested. As an example, a decision module can be implemented with a task planner or it can be controlled by the experimenter in the style of a Wizard of Oz study. These implementations should be exchangeable within the framework, irrespective of perception, action or metric module implementations.

This framework targets HRI researchers as the main users and hence, it should be easy to use in terms of study setup as well as data collection. This is why we introduced a metrics module. In HRI, metrics are often gathered via manual labelling of robot and human actions through analyzing video after the study is complete. In our system, we aim to make this process easier through an online labelling system that can provide an initial set of quantitative metrics after the study is complete. This is detailed further in Subsection IV-D.

IV. Framework Architecture

At a high-level, the framework architecture is decoupled into four main modules: perception, decision, action, and metrics. Figure 2 shows the architecture in block diagram form with short descriptions for each module. In the following subsections, each of the modules are described in further detail.

A. Perception Module

The perception module aggregates sensor input into usable data for the decision and action modules.

The implementation serves as an abstraction for ROS transform utilities, allows multiple tracking systems to be used simultaneously, and allows the user to specify additional points of interest on objects. The module has a plugin design

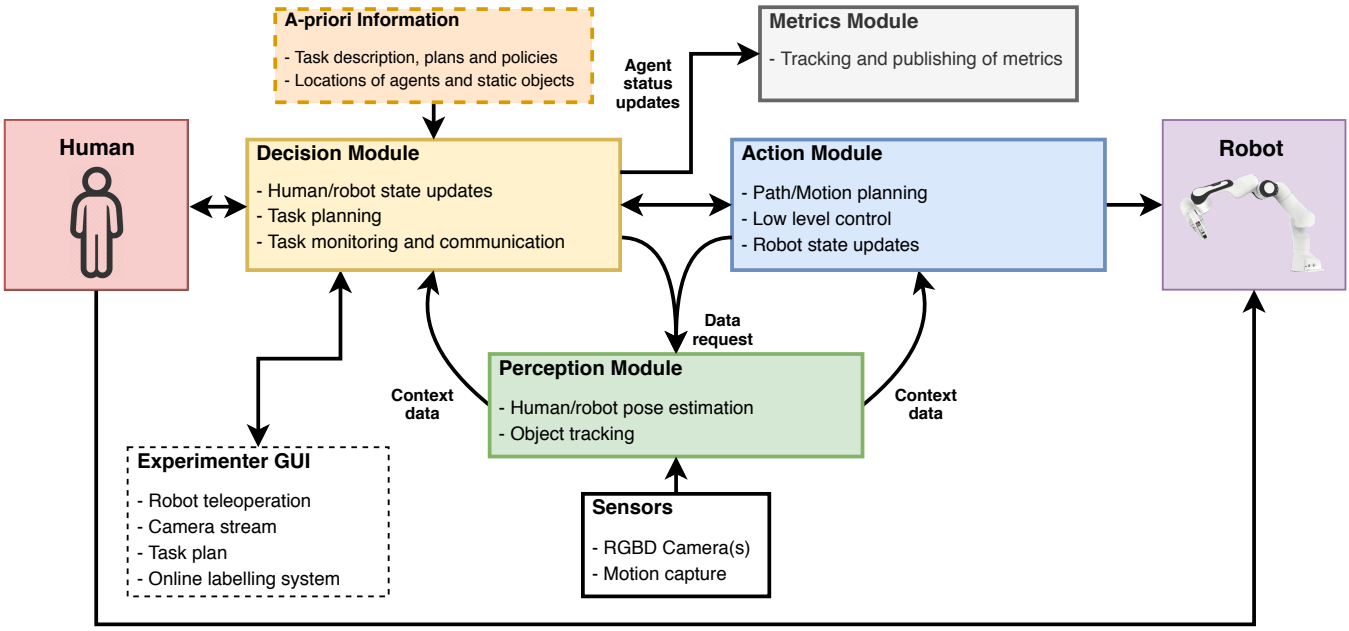

Fig. 2: Block diagram for the proposed HRI framework.

which allows it to be used with different third-party tracking systems. This includes object tracking and human/robot pose estimation, such as implementations of ArUco tracking, VICON motion capture and image segmentation systems for common HRI tasks.

The module also spawns ROS services that can be queried with object identification numbers to retrieve object poses or body parts to obtain poses of human participant limbs. Modules wanting to communicate with and obtain object poses from the perception module send requests containing an integer corresponding to the object ID, a string indicating which tracking system the object is in (e.g., "ArUco", "MOCAP", "Segmentation") and a string indicating a query point (e.g. `"marker_center"`, `"grasp_point"`) as shown in the input-output diagram in Figure 3.

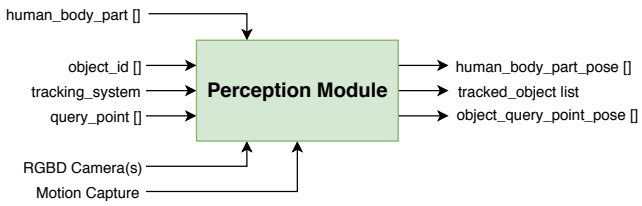

Fig. 3: Perception module Input-Output Diagram.

Object query points, shown in Figure 4, are user-defined 6-DOF poses that are tracked relative to the base frame of the object. This gives researchers the ability to, for example, specify points at which objects can be grasped or locations for handovers to a human agent. These query points can serve to simplify implementation of the action module.

While initializing the perception module, users can provide 3D CAD models of the objects or describe them in bounding box form using primitive shapes such as boxes, cylinders or spheres. This information can be used, for example, to provide the robot with a virtual scene to do collision-checking during planning. Query points are also specified during the initialization of the module and can be stored for reuse through multiple study trials.

The only prerequisite hardware required for perception is an RGB camera. One of the many ROS packages that simulate off-the-shelf cameras (e.g., [17]) can be used in tandem with RVIZ and Gazebo to instantiate and simulate the perception module for testing without actual hardware.

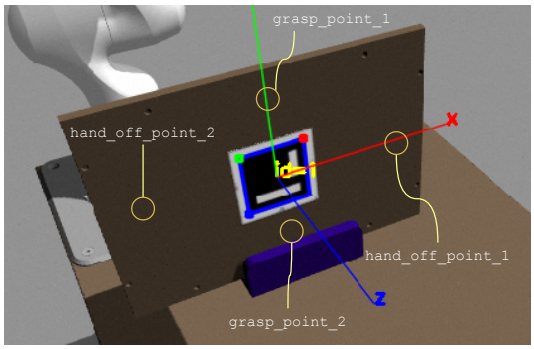

Fig. 4: An example of a set of query points on an object of interest shown in the Gazebo simulator.

B. Decision Module

The decision module is the task-level executive that breaks down the high-level goal into assignable sub-tasks for the agents in the study (i.e., the human and the robot). From the input-output diagram shown in Figure 5, the module takes as input the task and agent state information, such as the tracked

object and/or human poses from the perception module, for conditional updates within any behaviour framework such as hierarchical state machines or behaviour trees. The module also exposes ROS services for teleoperation input, human preemption requests and experimenter override requests. The module can also be tested in simulation.

The teleoperation service can take in a named object from the list of tracked objects in the perception module and an action (e.g., pick and pass, move to pose etc.) to make a call to the action module to perform a robot action. The researcher can encode these service calls as buttons in an experimenter-facing GUI to conduct a Wizard of Oz style study.

The human preemption request service enables the human participant to, e.g., stop the study if needed. It is up to the researcher to implement any preemption logic within their behaviour design.

Finally, the experimenter override request service allows experimenters to trigger conditional updates to their behaviour design implementation to, for example, mark an action as successful to move the user study forward. This can be especially useful in pilot studies where certain failures can be overridden to avoid having to reset the study entirely.

The interface also enforces the reporting of the "Robot state update" and the "Human state update". This update has three fields: a string indicating what action was performed and two floats for absolute start and stop time since the start of the study. These updates are used to calculate metrics for the user study.

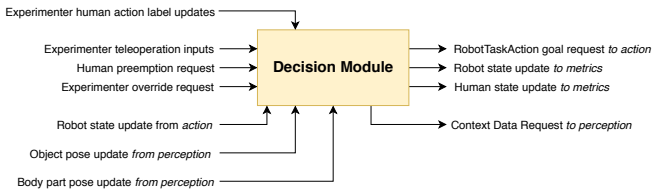

Fig. 5: Decision module Input-Output Diagram

All of these services are consolidated into an RQT dashboard and a reference implementation that can be customized to various study specifications is included with the framework.

An important note for this module is that for an accurate and semi-automatic calculation of metrics, all human and robot subtasks need to be encoded into the decision task plan and subtask start and stop times need to be reported to the metrics module. If a subtask that was not encoded into the task plan occurs during the study (e.g., a human participant performs an unanticipated action), the researcher can create a new label and indicate start and stop times of the subtask while the study is taking place.

C. Action Module

The action module takes requests from the decision module and executes them on the robot hardware. The current implementation uses the object tracking framework from the perception module to compute whether objects and/or the robot are in collision via MoveIt [18]. Predefined meshes or bounding box models are used to build collision models of the robot and objects.

For modularity and code reusability, an abstract `Robot` class with common robot functionality was created. This class can be configured to run using MoveIt to generate collision free trajectories or without MoveIt if the robot does not have a MoveIt implementation. This base Robot class has functions implemented for moving to 6-DOF poses, joint value targets and other common actions for HRI. However, some functions are left unimplemented for the researcher to complete for their particular robot. For example, opening and closing grippers are very robot dependent, based on the type of gripper and number of actuators. Figure 6 shows an example of how a researcher might extend the `Robot` class for a particular robotic arm such as the Franka Emika Panda Arm and add arm specific functions to suit their experimental setup.

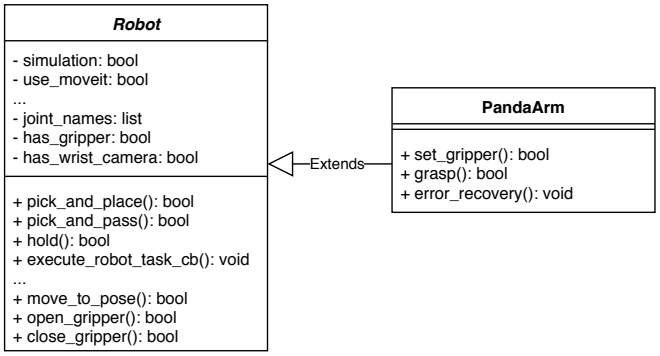

Fig. 6: Arm base class inheritance example.

In extending the base `Robot` class, the researcher must also implement a ROS `actionlib` action server for their particular robot to satisfy the action module interface. This action server should include the basic functionalities needed for the human-robot collaboration scenario such as move to pose, pick and place, pick and pass, pick and hold, hold, open gripper, close gripper, among others. These become the primitives available for the decision module when constructing the user study. Figure 7 shows the input-output diagram for the action module.

This module can also be simulated using RVIZ and Gazebo as long as the robot has a kinematic and dynamic model in URDF form.

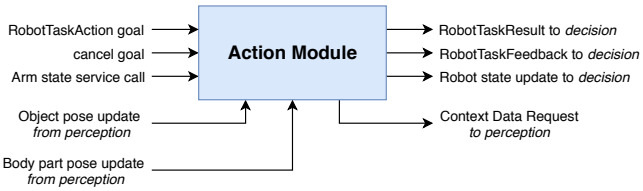

Fig. 7: Action module Input-Output Diagram.

D. Metrics Module

The metrics module aggregates information from the decision and action modules and publishes metrics related to the study. For the current implementation, only quantitative measures such as system and team performance are considered. The inclusion of metrics allows researchers to assess the impact that different module implementations and/or framework compositions have on the task, robot, and/or system (human-robot team) performance. The selected metrics are (in most cases) agnostic to the content of the interaction scenario. The metrics currently included in the proposed framework were derived from [6] where a minimal turn-taking model for human-robot collaboration is proposed. The metrics are described below:

1) **Team effectiveness:** percentage of the task that was achieved with the designed autonomy. This can be tracked via the number and duration of human interventions.
2) **Team efficiency:** amount of time required by the human-robot team to successfully complete the study.
3) **Agent idle time: (R-IDLE and H-IDLE):** percentage of total time an agent (robot or human) is not executing an action.
4) **Concurrent activity (C-ACT):** percentage of total study time during which both human and robot were active simultaneously.
5) **Functional delay (F-DEL):** the accumulated time between the completion of one agent's action and the beginning of the other agent's action, calculated as a percentage of the total task time.

As explained in Subsection IV-B, the "Arm state" and "Human state" updates contain the absolute start and stop times for each human and robot subtask. From the H-IDLE, R-IDLE, C-ACT and F-DEL equations listed in [6], this is all that is needed to compute these objective human-robot fluency metrics. As such, the metrics module can be customized and extended with new quantitative metrics that require as input the duration of the human and/or robot actions.

Fig. 8: Metrics module Input-Output Diagram.

E. Setup

Before usage of the framework, a few setup steps need to be completed. First, the researcher must identify what needs to be perceived in their experiment and instantiate a tracking system for important objects in the environment. This includes attaching tracking markers to the objects if needed and providing 3D representations of the objects for obstacle avoidance while planning. Calibration also needs to be performed if using a camera on the end-effector or externally in the environment using an implementation such as easy_handeye: TF / VISP Hand-Eye Calibration [19].

Second, the researcher should identify the set of robot actions needed for the user study and extend the base `Robot` class for their own robot and implement any task specific functionality that is not already provided. They also need to instantiate the robot action server that takes requests from the decision module and performs actions on the robot.

Third, the researcher will need to design the decision algorithm and encode all robot and human behaviours into their task plan and ensure they have customized the experimenter-facing GUI with any study-specific teleoperation or annotation inputs.

Finally, a set of metrics to be reported at the end of the study needs to be decided upon and implemented if not already available in the framework.

V. Validation and Evaluation of Framework

To illustrate the framework, a task similar to [20] was used, where a robot and human participant assemble a wooden toolbox. As described in Section III, design specifications were introduced to guide the development of the framework. We demonstrate how these specifications were met through this collaborative task scenario. In our task, a human and robot must collaboratively assemble a 6 piece toolbox as shown in Figure 9.

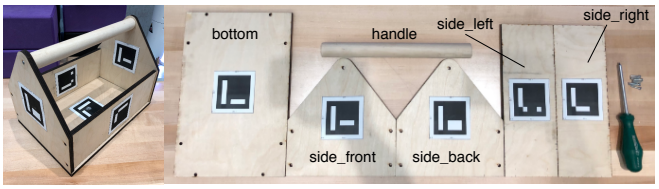

Fig. 9: Assembled toolbox and parts with attached ArUco markers.

Similar to the description in [20], the different parts have to be:

1) Passed to the human
2) Positioned for assembly
3) Attached using screws

To implement the task within the framework, we devised two implementations for each module. This was done to show the interoperability of the framework. Table I shows the various implementations labeled "A" and "B" for each module. Table II shows the three trials that were run and the associated module implementations used. Figure 10 shows the task setup.

For implementation A of the perception module, we used ArUco markers attached to both sides of all of the parts as shown in Figure 9. Three dimensional models of the parts were used as obstacles to aid the action module for motion planning and to also locate the parts for pick and pass actions. Points of interest, such as grasp points and handover points, were added to the parts as query points. OpenPose [21] was used to track the human agent's right arm for handoffs and the ArUco markers were actively tracked throughout the task.

	Implementation A	Implementation B
Perception	ArUco based object tracking, OpenPose for human pose estimation	Predefined 6-DOF poses for object and handoff locations
Decision	Robot and human tasks encoded into a behaviour tree, experimenter labels human task start and stop times	Experimenter teleoperates the robot and labels human task start and stop times
Action	Online collision-aware motion planning	Pre-planned motions to all objects and handoff position

TABLE I: Toolbox task module implementation summaries.

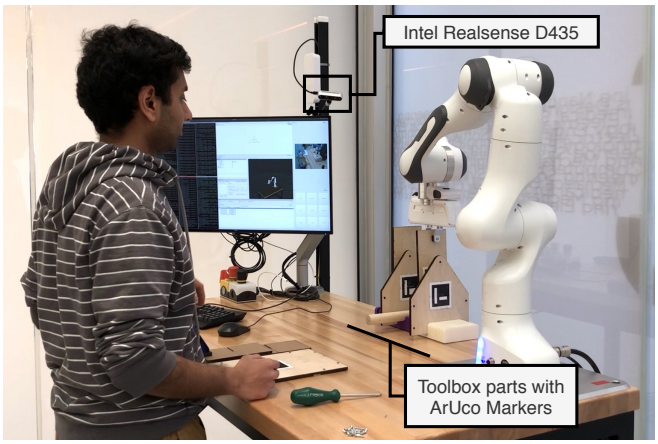

Fig. 10: Toolbox task setup.

Trial	Perception	Decision	Action
1 - Autonomous	A	A	A
2 - Teleoperated	B	B	B
3 - Semi-Autonomous	A	B	A

TABLE II: Toolbox task module versions for each trial.

For implementation A of the action module, MoveIt was used to plan collision-aware robot trajectories during the task. MoveIt's capability to attach and detach objects to the robot gripper was employed when picking and passing objects to the human participant. The PandaArm class from Figure 6 was used to spawn an action server with implementations for all of the basic motion primitives. Some of these motion primitives can be seen in Figure 11. For the decision module, implementation A encoded all human and robot actions into a behaviour tree (using the pi_trees [22] package) and the experimenter was in charge of labelling human action start and stop times during the experiment.

For implementation B of the perception module, static locations of objects were predefined along with a 6-DOF handover position for passing parts to the human. For implementation B of the action module, offline motions were computed to all of the static objects and the handover location. For the decision module, implementation B allowed the experimenter to teleoperate the robot by selecting which part and what action to perform. They were also in charge of labelling human action start and stop times.

The main difference between each set of implementations was the level of robot autonomy and how automated each of the modules were. The set of "A" implementations provide more autonomy than the set of "B" implementations.

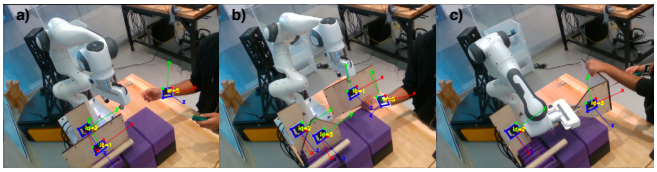

Fig. 11: a) Robot retrieving a part from the holder b) Robot performing a handover to the human c) Robot holding a part in place for the human

Trial 1, labelled "autonomous", consisted of all modules operating under implementations A. Each task for both the human and robot were encoded into a behaviour tree and there were no human participant interventions or experimenter overrides.

For trial 2, labelled "teleoperated", the experimenter was free to teleoperate the robot and select parts and actions for the robot to perform. All modules ran using implementations B. In this trial, there was an experimenter intervention due to the robot knocking a part out of place while in motion, causing a failed grasp.

For trial 3, labelled "semi-autonomous", the action and perception modules ran using implementations A while the decision module was teleoperated by the experimenter and ran using implementation B. In this trial, there were no experimenter or participant interventions.

Table III lists the metrics gathered from each trial of the toolbox task. For both the autonomous and semi-autonomous tasks, effectiveness was 1.0 because the full study was achieved using the designed autonomy and there were no participant interventions. The autonomous case had the best efficiency with a study completion time of 320.71s. Swapping the decision module to a teleoperated version for the semi-autonomous case had an effect on the efficiency of the study but not so much on the agent idle times and functional delay.

Through the teleoperated trial we begin to clearly see the effect that different module implementations have on the overall study metrics. The effectiveness was 0.95 due to one experimenter intervention, and this study variant took the longest to complete. It also had higher human and robot idle times. This is expected due to the fact that the experimenter was operating the robot and induced some delay in triggering actions. This is reflected in the functional delay metric, which was higher in the teleoperated trial compared to both the autonomous and semi-autonomous cases. The higher study completion time and lower effectiveness can also be attributed to the fact that this variant of the study had no

active object tracking and instead relied on predefined object locations. Hence the effect of different perception, action and decision module implementations is directly reflected in the study metrics.

Trial	Effect.	Effic.(s)	H-IDLE	R-IDLE	C-ACT	F-DEL
1 - Autonomous	1.0	320.71	0.31	0.44	0.29	-0.09
2 - Teleoperated	0.95	410.51	0.34	0.51	0.26	0.06
3 - Semi-Autonomous	1.0	352.3	0.31	0.48	0.28	-0.05

TABLE III: Toolbox Task Study Metrics

As explained in [6], F-DEL can be negative if there are many overlapping actions. This was the case for the autonomous and semi-autonomous trials because the human was always performing a task prior to collaboration with the robot.

Please refer to the accompanying video[2] to see an example of the toolbox task study with a human participant.

VI. Discussion

In this work, we introduced a standardized implementation framework for human robot interaction studies through which HRI researchers can implement, revise and conduct experiments, as well as test different module implementations. The system consists of four main modules, perception, decision, action, and metrics which can all have varying implementations as long as they satisfy the simple module interfaces outlined. To make studies more easily comparable, we integrated an interface for calculation of several metrics related to human-robot fluency. We outlined how a new researcher would set up their own task within the framework and how they could extend module functionality with new features. In the exemplar HRI scenario detailed in Section V, we were able to easily swap different module implementations in a *modular* and *interoperable* fashion and evaluate the effects on the study metrics. The framework facilitates module testing in the full context of an HRI experimental setup to assess how different implementations affect task and study performance.

A limitation of our current work is that it is geared towards HRI studies with one human and one robot. Another limitation is that the perception module interface mainly caters to vision-based systems. Finally, for the tracking of metrics, human actions need to be encoded into the study or be labelled manually by the experimenter while the study is in progress.

Future work includes extending the perception module with new modalities such as speech recognition. We also hope to develop an interaction module to extend the framework for use cases such as social and cognitive HRI user studies. Finally, we intend to expand the list of objective metrics while also automating the gathering process further by, for example, utilizing activity detection algorithms to automatically calculate agent idle time.

[2]Also available at https://youtu.be/0KglQmCkpAQ.

References

[1] A. Ajoudani, *et al.*, "Progress and prospects of the human–robot collaboration," *Autonomous Robots*, vol. 42, no. 5, pp. 957–975, 2018.

[2] A. Khan *et al.*, "Robots in healthcare: A survey," in *Science and Information Conference*. Springer, 2019, pp. 280–292.

[3] S. Zeylikman, *et al.*, "The HRC model set for human-robot collaboration research," *CoRR*, vol. abs/1710.11211, 2017.

[4] A. Murali, *et al.*, "Pyrobot: An open-source robotics framework for research and benchmarking," *CoRR*, vol. abs/1906.08236, 2019.

[5] A. Steinfeld, *et al.*, "Common metrics for human-robot interaction," in *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-robot Interaction*, ser. HRI '06. New York, NY, USA: ACM, 2006, pp. 33–40.

[6] G. Hoffman, "Evaluating fluency in human–robot collaboration," *IEEE Transactions on Human-Machine Systems*, vol. 49, no. 3, pp. 209–218, June 2019.

[7] J. Lindblom *et al.*, "Current challenges for ux evaluation of human-robot interaction," in *Advances in ergonomics of manufacturing: Managing the enterprise of the future*. Springer, 2016, pp. 267–277.

[8] K. Belhassein, *et al.*, "Towards methodological principles for user studies in human-robot interaction," in *Test Methods and Metrics for Effective HRI in Collaborative Human-Robot Teams Workshop, ACM/IEEE International Conference on Human-Robot Interaction*, 2019.

[9] M. Quigley, *et al.*, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[10] A. A. Malik *et al.*, "Digital twins of human robot collaboration in a production setting," vol. 17, 06 2018.

[11] P. Tsarouchi, *et al.*, "ROS Based Coordination of Human Robot Cooperative Assembly Tasks-An Industrial Case Study," *Procedia CIRP*, vol. 37, pp. 254 – 259, 2015.

[12] I. Lane, *et al.*, "HRItk: The human-robot interaction ToolKit rapid development of speech-centric interactive systems in ROS," in *NAACL-HLT Workshop on Future directions and needs in the Spoken Dialog Community: Tools and Data (SDCTD 2012)*. Montréal, Canada: Association for Computational Linguistics, June 2012, pp. 41–44.

[13] A. Clodic, *et al.*, "Shary: a supervision system adapted to human-robot interaction," in *Experimental Robotics*. Springer, 2009, pp. 229–238.

[14] C. Paxton, *et al.*, "CoSTAR: Instructing collaborative robots with behavior trees and vision," *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 2017, available as arXiv preprint arXiv:1611.06145.

[15] T. Fischer, *et al.*, "iCub-HRI: A software framework for complex human-robot interaction scenarios on the iCub humanoid robot," *Frontiers in Robotics and AI*, vol. 5, no. 22, pp. 1–9, 2018.

[16] T. Fong, *et al.*, "The human-robot interaction operating system," in *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-robot Interaction*, ser. HRI '06. New York, NY, USA: ACM, 2006, pp. 41–48.

[17] S. Missri, "realsense_gazebo_plugin," https://github.com/SyrianSpock/realsense_gazebo_plugin, 2019.

[18] I. A. Sucan *et al.* MoveIt. [Online]. Available: https://moveit.ros.org/about/

[19] E. Marco, "easy_handeye: TF \VISP Hand-Eye Calibration," https://github.com/IFL-CAMP/easy_handeye.

[20] M. Toussaint, *et al.*, "Relational activity processes for modeling concurrent cooperation," in *(ICRA 2016)*, 2016.

[21] Z. Cao, *et al.*, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *CoRR*, vol. abs/1812.08008, 2018. [Online]. Available: http://arxiv.org/abs/1812.08008

[22] P. Goebel, "pi_trees," https://github.com/pirobot/pi_trees, 2019.