Investigating the Effects of Robot Interactivity on User Experience

Edward Stables (01220379), Josh Jennings (01221510), Jack Waller (01217931)
Fangfang Hu (01204764), Oliver Spillet (01195497), Alex Prescott (01191642)

***Abstract*—In this project, a study is to be conducted into the effects on user experience from varying the amount of interaction a robot has with a user. This is done through a custom robot with features that allow it to engage with the user with different types of actions. The robot will receive user feedback through an integrated rating scale; this will allow for the evaluation and analysis of these interactions. In this report, the design of the robot is explained in detail. Robot features include motion, facial detection and voice recognition, with additional subsystems for the artificial intelligence (AI), visual display and a status monitor.**

I. Introduction

Robots with the capability to interact with human users have been the subject of numerous past studies, especially in terms of the acceptance of such robots by human users. For instance, a 2010 study investigated the feature design effects of a service robot on user perceptions and emotional responses, whereby the robot was tested with twenty-four elderly participants [1]. However, a 2017 study on social acceptance of a long-term autonomous robot at a care hospital found mixed results with regards to the social acceptance of robots by human users [2]. By investigating the effects of having different types and combinations of user interaction by a robot, there is potential to find an optimal compromise between functionality and social acceptance of robots by human users. Hence, in this project, a robot was designed to investigate these effects.

II. Existing Material

Results from the aforementioned 2010 study indicated that anthropomorphic and interactive features to service robots promoted positive emotional responses. The previously mentioned 2017 study also stated that a robot with meaningful communication abilities and cues enhancing the predictability of its behaviour increased the social acceptance of the robot by human users. Hence, the overall design of the robot was determined using the fundamental interactions expected of an anthropomorphic, interactive robot: 1) movement, 2) speech capabilities, and 3) visual feedback.

III. Design and Current Implementation

A. Mechanical Design and Control

The mechanics of the robot are designed to execute physical motions, namely to angle itself towards the current user and hence, track any of the user's movements. It would also enable the robot to engage in the expression of its current state ("emotion") through physical movements. This could be a movement whereby the robot repeatedly changes the positions of some its parts to a rhythm, and thus can be interpreted by the user to be 'dancing'. This physical movement constitutes a type of interaction with the user.

For the movement control of the robot, a simple morphology comprised of a fixed base and moving upper part (the "head") was chosen. This was inspired by Keepon, a small robot designed by Hideki Kozima to study social communication through interaction [3]. To implement the moving "head" of the robot, a Stewart platform was chosen as it allows for six degrees of freedom with few mechanical components. It also allows the centre of the robot to be kept clear of moving parts, enabling umbilical wiring between the "head" and base of the robot. The control of the Stewart platform is based on the movement of the "head" in the x, y, and z directions indicating the spatial position of the "head", as well as pitch, roll and yaw indicating the axes of rotation. Following which, these are converted into angles for each rotary servo motor using the inverse kinematics in an online simulator [4] and a previous publication on the mathematics behind the Stewart platform [5].

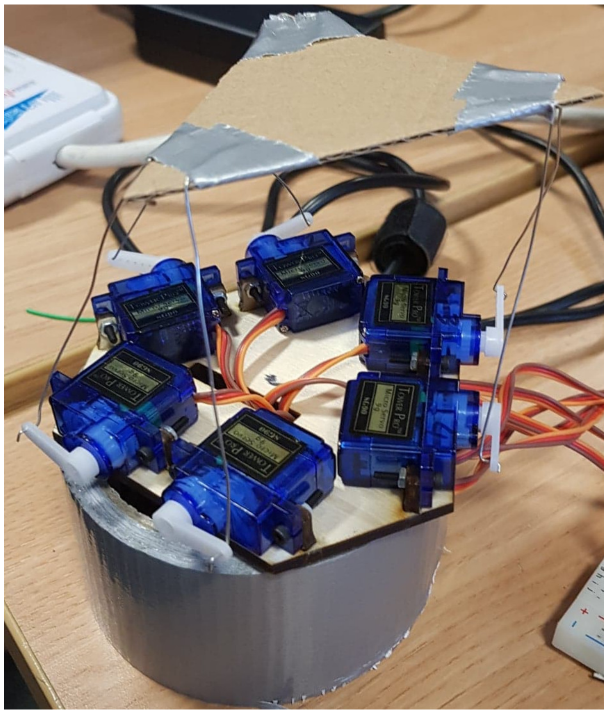
Fig. 1. Initial Platform Used For Testing Stewart Platform Library

The program was then ported to C++ and modified to run on the Teensy development board, which is used for kinematic control. It was then rewritten into a library, allowing more concise control of the robot's movements. A preliminary platform was assembled as shown in Figure 1. This allowed for basic movement testing using the library.

B. Software Design

A Publisher-Subscriber system was designed, utilising a Publisher-Subscriber pattern illustrated in Figure 2.

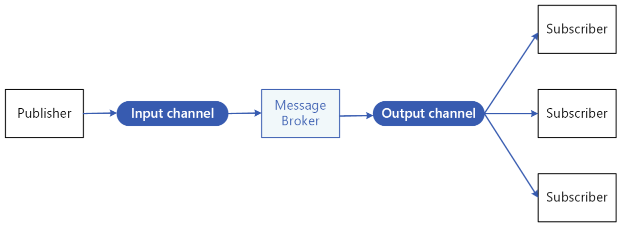

Fig. 2. Components of the Publisher-Subscriber pattern used [6].

This system has a simple interface, allowing easy integration of each individual subsystem. By using a lightweight system instead of ROS, the development process of each constituent subsystem was made more efficient. This was because each member of the team could focus solely on the subsystem which they were responsible for and only a single member was required to build the messaging system and integrate the various subsystems.

Each subsystem was developed in Python. This is because Python allows for fast prototyping and preliminary development. It was decided that this ease and efficiency of development justified the compromise on computational efficiency compared to other languages such as Java or C. Furthermore, as each subsystem does not rely on high computational speeds, Python was sufficient for development. However, it is possible that with further development and evaluation of the robot's functionalities, a transition to higher performance systems such as ROS could be needed.

The software architecture is shown in Figure 3.

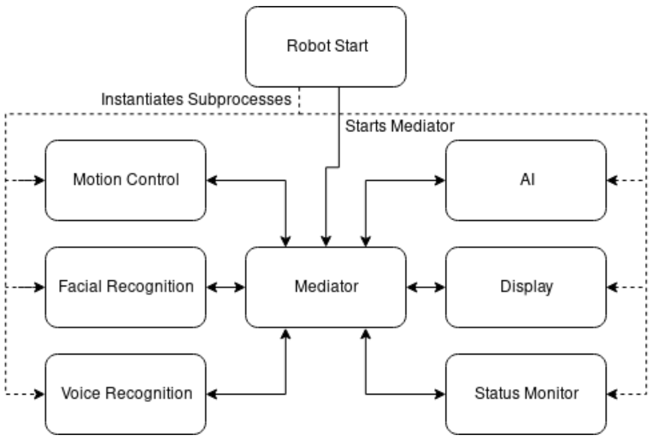

Fig. 3. Overview of software architecture. Each of the six subsystems runs in its own process.

As illustrated, each subsystem is started in a new process, with a single central process (the "mediator") receiving and sending messages between subsystems. This use of multiple processes avoids Python's limitation of running on a single process[7] and makes full use the quad-core processor in the Raspberry Pi 4[8] used in the robot.

Upon creation of a subsystem, it will fork itself into a new operating system process and return a pipe to the mediator, allowing process-safe communication between the subprocesses. While the pipes allow for messages to be sent in a safe manner between processes, they themselves are not thread-safe within processes[9]. Therefore the subsystem design provides an abstraction layer between the pipe and the subsystem, making both sending and receiving messages thread-safe.

C. Hardware Software Interface Design

The interface between the hardware and software components in the robot is handled by an Arduino. The purpose of this subsystem is to take input from the Pi regarding required movement and robot state, then use these instructions to control the platform and the state indicator. Instructions are sent from the Raspberry Pi 4 to the Arduino in JSON. This is because JSON is very flexible with the format of data that can be sent. A JSON command is read into the Arduino serially as a string with a format similar to that of the instruction shown in Figure 4. Another benefit of choosing JSON format is the pre-existing library, ArduinoJSON [10]. This library makes it very easy to parse JSON instructions.

```
[
    {
        "state": 3,
        "instr": 1,
        "moves": [[0, 0, 0, pi/4, pi/4, pi/4],[1, 1,
             1, pi/2, pi/2, pi/2]
    }
]
```

Fig. 4. Example JSON Format

There are three possible instructions: reset, movement pattern and offset correction. "Reset" returns the Stewart platform to its neutral position. This is done using the Stewart platform library as discussed previously in III-A. A movement command is sent to the platform with x, y, z translation, pitch, roll and yaw all set to zero. This command uses instruction op code 0. Movement patterns use op code 1. This means that the JSON instruction will contain an array of moves for the platform to make. Finally, offset correction is op code 2. For this command, the JSON instruction will contain an array of two values between -1 and 1 representing x and y offset. This format was chosen as it means that the JSON instructions can vary in form and new instructions can be added easily.

In addition to the movement required, each command will have a state field. This is so that the Arduino can update the state indicator.

In future development, movement patterns could be preprogrammed into the Arduino such that the Raspberry Pi need only send a single instruction, for example, an action such as "dance" or "nod".

The Arduino acts as a hardware controller for the Stewart platform. This setup was selected as it makes the main processor (the Raspberry Pi 4) more versatile. In theory, the processor could be placed in a different robot by making changes only to the specific hardware controllers, while no changes have to be made to the core programming.

D. AI Design

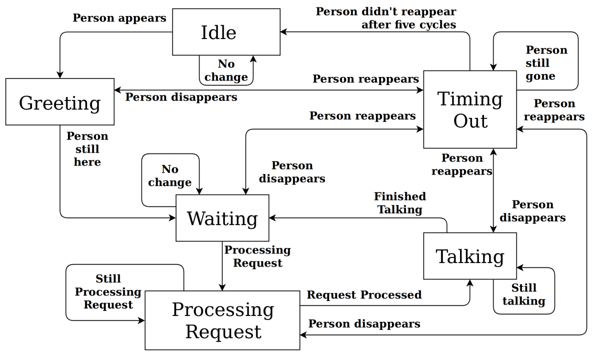

Fig. 5. State diagram used by the AI subsystem to control behaviour.

The AI subsystem determines how the robot functions and what happens at any given point. It uses a system of flags that represent important information, such as whether there is a person in the robot's line of sight or whether the robot is processing a request. These flags are used by a state machine that changes states based on the current state and which flags are set. Since the robot can only receive or process commands if it is in a specific state, this avoids issues such as the user asking a question when the robot is not meant to be listening to a new command. This could be the case when the robot has not yet completed a different command.

Furthermore, the use of a custom state machine makes it easy to add new states and flags to the subsystem. For example, a new behaviour can be introduced with the addition of a single state class and defining transitions conditions into and out of this state.

E. Visual Display Interface

The display interface serves two functions in the design. It gives the robot an obvious focus point for the user when interacting, while the touch interface acts as a fallback for user input after if voice recognition fails to comprehend the user's speech.

Whilst speech is intended to be the primary method of user interaction with the robot, it is possible that a situation could arise in which the robot will be unable to decipher what the user has said. For instance, this could happen in environments with a lot of background noise which could result in the microphone not picking up the user's speech. In these situations, the interaction could break down into a long loop of the user being asked by the robot to repeat themselves. In order to avoid this, the robot will have the ability to resort to asking the user to interact with a touch screen. This will allow the robot to be able to interpret interactions correctly and avoid responses that are out of the scope of the robot's functionalities.

The addition of "eyes" to the robot will also increase the extent to which users engage with the robot [11]. These "eyes" will be used in conjunction with the motion control subsystem to follow the user currently interacting with the robot. These "eyes" can also exhibit varying levels of expression of emotion, allowing the evaluation of how emotional expression by a robot could affect user experience.

The design decision was also made to not include a "mouth" for the robot, despite the capability of the robot to "speak" to the user. This was because the animation of a mouth on the robot was deemed to be of a low priority compared to the remainder of the design of the robot.

The visual display interface also includes a ratings scale with which the user can give feedback. This will be used in the experiment, however its exact implementation is subject to change as the robot improves.

In the current state, the user interface consists of two screens. One screen is the user interface itself, while the other screen comprises of a placeholder debugging menu for use in development, Fig. 6. The user interface consists of a pair of eyes which have the ability to be controlled by other subsystems. The debugging menu allows the testing of features in the initial prototyping stage and will be removed for the final version of the robot.

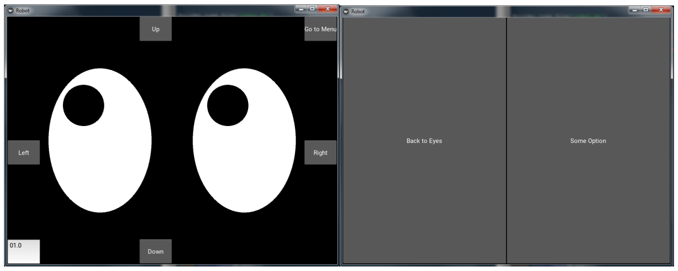

Fig. 6. User interface screens for the robot. Left: Main UI Right: Debugging Menus

Following the current stage of development, the visual display interface will be integrated with the rest of the system while ensuring that all interactions between the subsystems work as intended. Increasing the range of ways in which the visual interface can be used as an expressive tool will also allow the robot to be used to study how varying levels of expressiveness can change the user experience. The interface for receiving user feedback is also yet to be implemented.

F. Voice Recognition and Generation

In the voice recognition subsystem, the first step was to extract Python strings containing words from input audio from the user's speech obtained through the microphone in the robot. This was done by using an existing Python package, Pocketsphinx [12], which is built for low-resource platforms such as the Raspberry Pi. Currently, the voice recognition subsystem works on keyword spotting whereby it picks up keywords in the user's input speech to the robot to identify the intent of the user. Upon identifying the relevant functionality as required, the information is passed to the relevant subsystems through the "mediator" in JSON.

However, keyword spotting is not a robust method for correctly identifying user intent, especially if more complex speech is involved. Hence, following the current prototype build of the voice recognition subsystem, statistical topic modelling can be used to identify relevant "topics" [13], which in the case of the robot would be the functions to be executed. This can be done by assigning probabilities to identified words in the given text derived from the user's speech, then matching these probabilities to a pre-defined set of functions the robot can execute. As there is a limited number of functions executable by the robot, such an approach is thus lightweight and feasible while also being more robust than keyword spotting.

G. Facial Detection

An important aspect of design of the robot is the ability to 'look at' users who are interacting with it. This takes the form of the eyes on the display, and moving the robot's "head" to track people's faces. Face tracking is performed with OpenCV, specifically the built-in pre-trained model for full-frontal face detection.

This model offers relatively reliable detection of multiple faces in the frame. It reports features such as the number, position and size of the faces without requiring significant computational power. Early tests show that the Raspberry Pi 4 should be capable of running this subprocess alongside the other systems.

An assumption which was made about user interaction with the robot was that person interacting with the robot will have the largest face in view by the camera in the robot. Therefore, this subprocess will report the position of the face with the largest size in view. This is then normalised to a pair of values in the range $[-1, 1]$, allowing the subsystem to be reliable regardless of the field of view or resolution of the camera being used in the robot. These positions are then reported to the visual display interface and the hardware controller.

The visual display will then use the position to move the eyes to the correct calculated position, to indicate to the user that the robot has noticed them. The "head" of the robot moves in conjunction with the eyes to further emphasise the action. As the Stewart platform has more freedom in movement than the eyes on the display, it will use the positions as an input to a PID loop, in which it will attempt to centre the face of the user in the camera as explained in Section III-C.

IV. Experiment

A. Hypothesis

The hypothesis for this study is that an increased level of interactivity of the robot will result in improved user experience as measured by a user feedback rating scale.

B. Experiment

For the preliminary rounds of experimentation, the robot be placed in the foyer of the Electrical and Electronic Engineering department, which is an area of high traffic within the Imperial College London university campus. The robot will be programmed to follow of a schedule with various rounds of experimentation with different levels of interactivity available as shown in Figure **??**. With each new user identified in each round of experimentation, feedback from the user will be collected using a rating scale displayed on the robot's visual display interface. It is expected that the increased levels of interactivity will result in a greater number of response, potentially with a greater proportion of positive responses.

However, the novelty factor of such a robot may impact the accuracy the results. To counteract the effects arising from the novelty factor, the experimentation period will be several days long. This is intended to better familiarise the users with the robot, thus being less likely to be influenced by the novelty factor.

	Levels of Interaction	Round 1	Round 2	Round 3
Motion	Following Faces		X	X
	Emotion-based Movement			X
Speech	Listening to User Speech		X	X
	Replying to User Speech			X
Display	Eyes Following Faces		X	X
	Eyes Expressing Emotions			X

TABLE I
CONFIGURATIONS USED FOR DIFFERENT ROUNDS OF TESTING

An experiment conducted over several days is also susceptible to external factors. Such an instance could be bad weather, which could result in lower user participation due to decreased traffic flow. Therefore, the rounds of experimentation with varying interactivity will be of short duration, to minimise the potential impacts of such external factors.

References

[1] T. Zhang, D. B. Kaber, B. Zhu, M. Swangnetr, P. Mosaly, and L. Hodge, "Service robot feature design effects on user perceptions and emotional responses," *Intelligent Service Robotics*, vol. 3, no. 2, pp. 73–88, Apr 2010. [Online]. Available: https://doi.org/10.1007/s11370-010-0060-9

[2] R. A. C. M. Olde Keizer, L. van Velsen, M. Moncharmont, B. Riche, N. Ammour, S. Del Signore, G. Zia, H. Hermens, and A. N'Dja, "Using socially assistive robots for monitoring and preventing frailty among older adults: a study on usability and user experience challenges," *Health and Technology*, vol. 9, no. 4, pp. 595–605, Aug 2019. [Online]. Available: https://doi.org/10.1007/s12553-019-00320-9

[3] "Keepon pro," Beat Bots. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

[4] T. Hersan, "Stewart platform," Oct 2017. [Online]. Available: https://www.instructables.com/id/Stewart-Platform/

[5] Wokingham U3A Maths Group, "Stewart platform inverse kinematics." [Online]. Available: https://cdn.instructables.com/ORIG/FFI/8ZXW/I55MMY14/FFI8ZXWI55MMY14.pdf

[6] *Publisher-Subscriber pattern*, Microsoft Corporation, July 2018. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/publisher-subscriber

[7] *Thread State and the Global Interpreter Lock*, Python Software Foundation. [Online]. Available: https://docs.python.org/3/c-api/init.html#thread-state-and-the-global-interpreter-lock

[8] "Raspberry pi 4 tech specs," Raspberry Pi Foundation. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

[9] *Exchanging objects between processes*, Python Software Foundation. [Online]. Available: https://docs.python.org/3.7/library/multiprocessing.html#exchanging-objects-between-processes

[10] B. Blanchon, "Arduino json library." [Online]. Available: https://arduinojson.org/

[11] K. Kompatsiari, F. Ciardo, V. Tikhanoff, G. Metta, and A. Wykowska, "It's in the eyes: The engaging role of eye contact in hri," *International Journal of Social Robotics*, Jun 2019. [Online]. Available: https://doi.org/10.1007/s12369-019-00565-4

[12] "Cmusphinx documentation." [Online]. Available: https://cmusphinx.github.io/wiki

[13] S. Li, "Topic modeling and latent dirichlet allocation (lda) in python," May 2018. [Online]. Available: https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24