CareBot Final Report

Bolun Zhang, Chun Iao Tai, Cong Chen, Endong Sun, Haochen Shen, Jiangnan Ye,
Miao Ju, Qihan Yang, Tianrui Chen, Wenqiang Lai, Ye Mao, Zhihan Yan

Imperial College London
Electrical and Electronic Engineering
South Kensington Campus
London SW7 2AZ, UK

***Abstract*—The current ageing society is experiencing the absence of welfare technology to give humanistic care to vulnerable groups, especially during the COVID-19 pandemic. Care Robots have strong potential in today's market, being utilised in hospitals and homes to provide the basic support and care for vulnerable people such as the elderly, children, and disabled. In this report, *CareBot* is proposed as an intelligent assistant and home companion which provides a series of auxiliary functions and human-robot interaction (HRI) for especially elderly in particularly home-care scenarios. CareBot conducts concrete behaviours based on the speech commands from users and the feedback from computer visions. This report primarily focuses on CareBot's proposed functionalities, initial technical work, and forthcoming experiment setup.**

***Index Terms*—home-care, elderly, human-robot interaction (HRI)**

I. Introduction

Population ageing is poised to become one of the most significant social transformations in the global world. With the demand for care professionals increasing, especially during the COVID-19 pandemic, society has to face the challenge of staff shortages in the care sector. CareBot was designed to address this problem by assisting the elderly in particularly home-care scenarios.

Similar to humans, CareBot could serve as a communication tool and offer accompanies to the elders to reduce their loneliness. In the meantime, the robot could monitor the health condition and help to ask for first aid in emergencies.

Moreover, CareBot has many advantages over humans. Firstly, care and professional services can often be too expensive for the elderly [1]. The CareBot will be significantly less expensive in the long-term than care service provided by a human while providing comparable services. Secondly, it could offer endless patience to the elders who might repeat the same question millions of times and prevent elder abuse. Thirdly, the human caregiver typically holds private information about the elders, and it is hard to protect the privacy. CareBot will never reveal the information of the user to others and sometimes it preserves dignity to elders. Last but not least, the robot will never "forget". A caregiver could forget to remind the elderly to take medicines or do exercises regularly. The CareBot could guarantee to remind the elders timely, to keep them healthy.

II. Hypotheses

CareBot can help the elderly to reduce loneliness and memorise the location of important objects. On top of this, an interactive humanoid robot is proposed to be a more effective caring service according to the following hypotheses:

Hypothesis 1 (H1 - Health Support) *CareBot can give correct health advice to enhance the well-being of the elderly.*

Hypothesis 2 (H2 - Item memorization) *CareBot is able to memorise location of interested objects and aids users in finding them.*

Hypothesis 3 (H3 - Fall Detection) *CareBot can detect falling events of users in their sights.*

Methods to validate and evaluate our hypotheses are discussed in Experiment Validation (VI).

III. Related Work

Assistive robots, especially those aimed at elderly care, have been a popular and rapidly developing area in recent years [2] (figure 1). Based on existing research and development, assistive robots for the elderly can be divided into two categories [3]: rehabilitation robots and assistive social robots. The latter can be further divided into: service type robots, which provide services like house cleaning (Roomba [4]), navigation (Pearl [5]), and support basic daily activities like dressed, eating, bathing; And companion type robots, designed for enhancing the psychological and physical health of elderly people (Paro, Jibo, Nabaztag and Buddy). [6]–[9]

One of the most popularly studied service type robots is the Nursebot project: Pearl. It provides two main functions: reminding people of daily activities like taking medicines and navigating through the nursing facilities [5], which is further expanded to other environments. Pearl only has a simple face to enhance the human-robot interaction experience.

Social robots Jibo, regarded as companion type personal assistance, has no capabilities of physical manipulation [8]. However, it has an obvious social presence and virtual face. Jibo can connect medical devices to monitor people and tell a doctor if detecting any health issue or alarm signals.

The seal robot Paro is another companion robot that targets elderly people. It is developed to simulate human-robot interaction and interactions between the elderly [6]. Paro has a soft

body and has eight actuators to provide movement, which is controlled by the behaviour-generation system, so that it can reduce the loneliness of elderly people.

The robotic puppy Aibo can play and interact with humans like a puppy [10]. It can know his name and respond to it when called, demonstrate many tricks according to commands and can find its charging station itself. It has been widely studied to assess the pet-type robot effects on the quality of life and symptoms of stress [11].

The Robotic home assistant Care-O-bot [12] is much more sophisticated and can realize abundant functions, incorporating service and companion [13]. It can provide services like setting the table, operating the electric appliance, and cleaning. It is equipped with a manipulator's arm to perform fetch and carry objects tasks such as books and medicines and holding and lifting tasks. Plus, it can provide mobility aid such as helping the elderly stand up from the bed and walking aid, including detecting obstacles and guiding to a target. It also can communicate with people and call an emergency when important signals are supervised. However, Care-O-bot is less humanoid and may be less acceptable by users for its huge size.

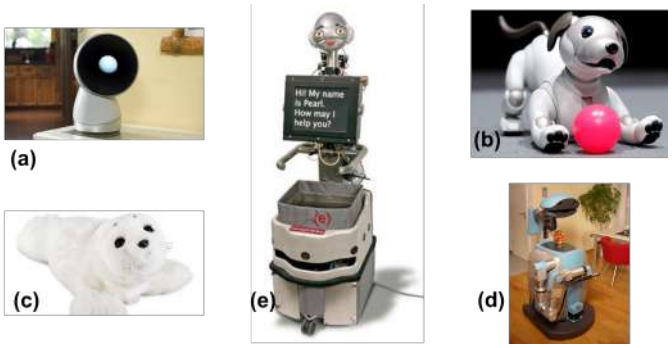

Fig. 1. Care Robots (a) Jibo (b) Aibo (c) Paro (d) Care-O-bot2 (e) Pearl

Most of the approaches introduced here address the problems of assistive robots for elderly, achieving the goals of companionship, assistance or rehabilitation. Typically, these approaches solve only **one problem** and are unable to cope with the complex and ever-changing situations in which older people live alone. Our proposed CareBot aims to perform **multiple tasks** to address a variety of problems. For example, solving the problem of older people who tend to forget the location of objects; proposing solutions to health problems for older people who are sedentary; or providing some daily weather reminders. In addition, most existing robots offer **limited interaction** with the user. In order to address the loneliness of elderly people living alone, CareBot increases the user experience by presenting a friendly **interactive systems**, such as dialogue and following systems. Moreover, in order not to scare the elderly, be suitable for placement in the home, and not to hurt the elderly during its tasks, CareBot is build on **Pepper**, which will be introduced more in the hardware design section.

IV. System Design

This section outlines the key states, modules, and functions in the autonomous care system designed for our CareBot. It utilises interactive systems to produce a dynamic human interactive process. This includes using the robot system to access information from the environment to form a decision-making system as well as demonstrating humanoid feedback to the user. The flowchart (figure 2) highlights how the control system achieves each of the key components by talking through the different stages of the decision-making from the start of the robot, to provide consistent care services.

There are two operation states: the background state and the interactive state:

A. Background state

The background state introduces the modules that are running regularly. There are two main functions:

- **Object detection:** CareBot will successively detect the presence of user-defined target objects in its sight and store the location in a database. This will help in finding them in the interactive state.
- **Fall detection:** The event trigger is designed for emergencies, for example, fall detection. The robot then applies a series of queries and actions aiming to offer appropriate help to the user, such as sending an urgent message to users' relatives.

B. Interactive state

The interactive state illustrates the details of human-robot interactions, which are mainly based on the dialogue between users and CareBot. Four types of basic functions are included in the Interactive state:

- **Following:** When the user give the command "Follow me", CareBot would track and follow the user until it receives the termination command "Stop".
- **Come here:** As the robot receives the "Come here" voice command from the user, it will try to find and approach the location of the user.
- **Chat:** When the user raises questions such as "what is the weather today?", CareBot could provide corresponding answers.
- **Find object:** When the user asked for the location of the object, CareBot would either approach the object and inform the user, or reply "I don't know" as an end.

V. Hardware Design

We did intensive research on hardware and software design to make the functions mentioned in the previous section realizable. Here, we dig more into the technical details and explain the hardware and software components we used in our project.

A. Hardware

The four main hardware are: robot, camera, microphone, and an extra processor (figure 3). They all integrated well with our system design and the software components.

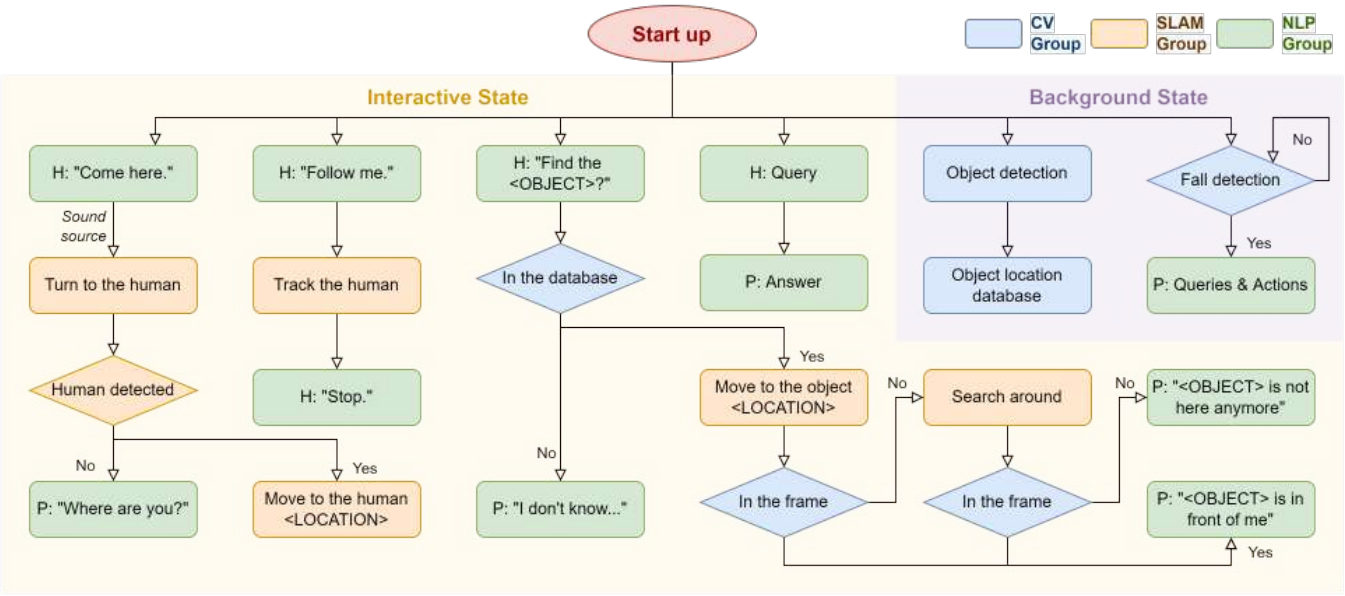

Fig. 2. Flow chart of the whole system design.

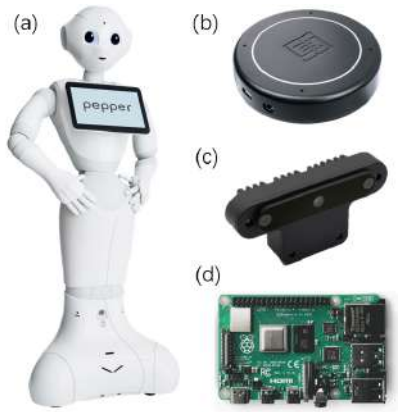

Fig. 3. Hardware used in the project: a) Pepper robot; b) ReSpeaker USB Mic Array; c) OAK-D Camera; d) Raspberry Pi 4.

1) Pepper: An appropriate choice of robot is Pepper. Pepper is the world's first social humanoid robot able to recognize faces and basic human emotions [14]. It was optimised for human interaction and is able to engage with people through conversation. Pepper is mounted with many sensors, which can satisfy our tasks including autonomous navigation: sonars, laser, and bumpers at the bottom of the Pepper could handle the obstacle avoidance. Besides, the design of the appearance of the robot is friendly to elders that it is not too high and strong to shock them.

2) ReSpeaker USB Mic Array: The ReSpeaker USB Mic Array is an out-of-box device with a well-designed acoustic structure. It uses 4 microphone arrays and 12 programmable RGB LED indicators to build voice-enabled applications such as Google Assistant and Alexa. There are several built-in functions we can use with the USB mic array, such as finding direction of arrival (DOA), voice activity detection (VAD), beamforming, and noise suppression.

3) OAK-D Camera: The OAK-D device is equipped with 3 cameras (e.g. one 12MP colour camera and a pair of 1MP stereo cameras) and an on-board Intel Myriad X VPU, which has 4 TOPS computational power. Such configuration allows us to manage computer vision tasks (e.g. object detection) on this single device in an end-to-end manner. In other words, the OAK-D device is able to record frames and input them into deployed computer vision models compiled with OpenVino, which can be processed with the VPU to obtain outputs for specific tasks. Thus, the portability and integrality of OAK-D make it our first choice of camera for the computer vision tasks in our project [15]. We used 2 OAK-D cameras in our project for fall detection and object detection respectively.

4) Raspberry Pi 4: The 2GB version of Raspberry Pi 4 Model B along with a battery pack serves as the controller and power supply for our camera and microphone. The final outfit is shown in figure 4.

VI. Software Design

To achieve all functions, besides additional hardware, methods such as Simultaneous Localization and Mapping (SLAM), Computer Vision (CV), Natural Language Processing (NLP), etc. are essential. Moreover, the Robot Operating system (ROS) integrates all the modules. The following part further breaks down the software details and explains how they are connected in ROS.

A. NAOqi Driver

Since Pepper itself are not capable of running ROS, we use the NAOqi Driver to bridge the Pepper NAOqi operating system and the Robot Operating System (ROS). After configuration, this driver brings up all the sensor data through

Fig. 4. Overall outfit of our system, including a Pepper robot, a ReSpeaker USB Mic Array and two OAK-D Camera attached on a wood frame, and a Raspberry Pi 4 stored in a bag

different topics and services in the ROS end, and allows us to use all the Naoqi APIs for Pepper.

B. Multi-Device Interaction

Running all ROS nodes on the Raspberry Pi is impractical, due to its limited computational power. Additionally, NAOqi APIs requires ROS with kinetic distribution, which only supports Python 2. Whereas most packages for computer vision are in Python 3, implying that various ROS distributions should be managed. The ideal solution is to allow several devices to connect with one another, with each device running a distinct ROS distribution and doing various tasks, as shown in figure 5. The first step is to install SSH server on the Raspberry Pi and PC respectively. Afterwards we add the other device's IP address and hostname to `/etc/hosts`. Finally, we utilise the Raspberry Pi as the master device to execute `roscore` and specified `ROS_MASTER_URI` and `ROS_HOSTNAME` in the *.bashrc* file.

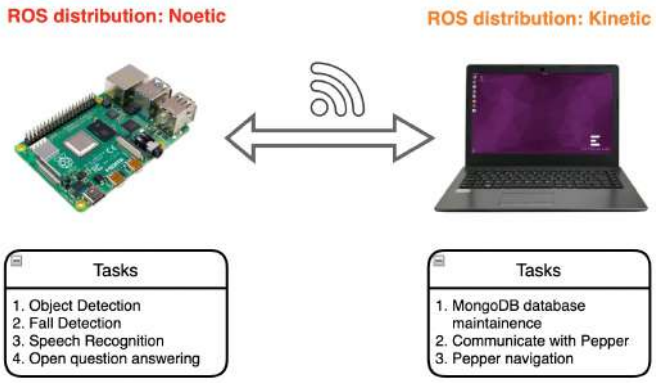

Fig. 5. Multiple devices for interaction and their corresponding ROS distributions and respective tasks.

C. Functionalities: Background State

Background state consists of functionalities that are constantly operating without triggering signals. As the main component of background state, computer vision helps CareBot monitor the real world through cameras constantly. An overall background state system design is shown in figure 6: The cameras conduct detection and sends both the class labels and pose estimations to the Raspberry Pi. Raspberry Pi, running two ROS nodes, keeps publishing the received information to the corresponding topics. In PC, a ROS node subscribes to the corresponding topic and writes the data to a database. It also provides a server to access the database. The following sections explain how to set up the camera and the CV models we used.

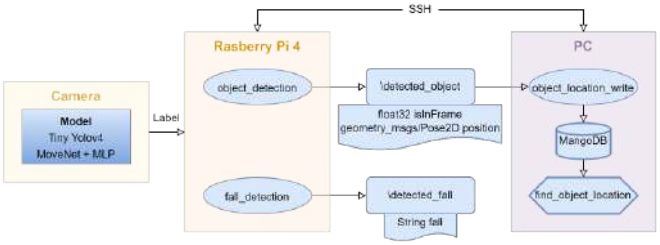

Fig. 6. Design architecture of the background state

.

1) Object detection node: As stated in the hardware section, the OAK-D camera undertakes on-board end-to-end computer vision tasks with the help of supporting frameworks, toolkits and deep learning models.

OpenVINO is a back-end framework for the optimization and the acceleration of the deployed AI models in the inference process [16]. It generates a cost-effective engine from models of prevalent platforms (e.g., PyTorch, TensorFlow) for the deployment on different devices, including the Intel VPU in OAK-D. Hence, OpenVINO is used in this project to optimize object detection models to be placed on the camera with less computational power.

The toolkit for the camera is Depth AI, which allows for convenient connection, configuration, and communication with OAK-D devices. It is used to control the camera pipeline and demonstrate images based on OpenCV. It also integrates the deploying process involved with OpenVINO.

The object detection model used in this project is Tiny-YOLO v4. It is a compressed variation of YOLO and pre-trained on the COCO dataset composed of 80 classes [17] (we use five: bottle, mouse, umbrella, remote, and cellphone). It has fewer parameters and offers faster inference (FPS: 39.8 [1]) with relatively high accuracy, making it suitable for real-time object detection (figure 7).

In terms of the overall procedure, the camera consecutively detects and returns the the results (e.g. the coordinates of bounding box corners, the labels of detected objects) to the Raspberry Pi. The Raspberry Pi is responsible for publishing a custom message to the `detected_object` topic. The message contains the following information: the flags that indicate the presence of object, the rotation angle between

[1]Depthai Model Zoo https://zoo.luxonis.com/

Fig. 7. Demonstration of CareBot performing real-time object detection.

robot and detected object, and the pepper location where it detected the object. To find the rotation angle, the horizontal pixel difference between the coordinate of bounding box center F_c and that of camera view center B_c is first obtained. This pixel difference is then multiplied with the vertical angular resolution $\Delta\theta$ of the camera to get the angle that the robot has to rotate. It is worthy to note that the vertical angular resolution is computed by dividing the vertical field of view (VFOV) by the total number of vertical pixels.

$$R = (F_c - B_c) \cdot \Delta\theta$$
$$\Delta\theta = \frac{VFOV}{P_v}$$

where F_c and B_c denotes the centre of the field of view (FOV) and bounding box respectively and $\Delta\theta$ defines the vertical angular resolution of the camera.

2) Database: To better manage the storage of object information, a MongoDB database is implemented in the PC. MongoDB is a source-available cross-platform document-oriented database program. MongoDB is classified as a NoSQL database, which is known for its high query speed and scalability [18]. The reasons for having a database include:

- Enabling long-term data storage;
- Allowing concurrent read and write operations;
- Providing an unique interface for any nodes requiring object information.

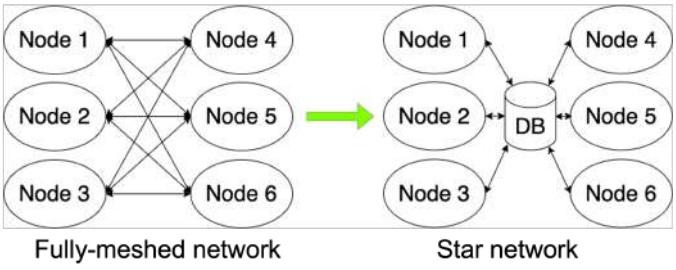

Fig. 8. Illustration of network topology simplification with database

Thanks to the scalability of MongoDB, the current database can be easily expanded to handle more data of different structures (e.g., memorising important events).

3) Fall detection node: The pose detection model used in this project is MoveNet, an ultra-fast and accurate model that detects 17 keypoints of a body to extract the skeleton-pose. The model consists of encoder, mapper, and decoder, where the encoder has five convolutional layers, the mapper has four fully connected layers and the decoder has five sets of one convolutional layer followed by one up-sampling layer subsequently followed by one max-pooling layer. The final output layer is convolutional layer to output skeleton-pose. Such a light model speeds up the real-time pose estimation [19].

Then we build an upper layer MLP to do the pose classification based on the features extracted from the 17 keypoints, where we set the threshold as 0.9 for classification, controlling the sensitivity of fall detection.

Similar to object detection, the human fall detection is also performed by using OAK-D camera. Models are transferred into `.blob` to match Depth AI toolkit, which can be recognised by OpenVINO. The cameras' embedded VPU combining with the framework OpenVINO accelerates the AI model computing, thus saving time and computing power to achieve real-time fall detection.

The camera keeps doing detection and returns the result (Fall or Normal) to the Raspberry Pi. Then the ROS node will publish a message to the `detected_fall` topic, which contains a variable tells the subscriber (the NLP master node) whether the subject has fallen or not.

D. Functionalities: Interactive State

Interactive state is composed of functionalities that require triggering signals. As shown in figure 2, most of the functions are triggered with dialogues. Hence, NLP plays a significant role in the interactive state. An overall interactive state system design is shown in figure 9: The microphone, doing sound source localization and speech detection, returns the sound source angle and detected word string to the Raspberry Pi. A `master_NLP` node will then takes different actions based on the word received. If any keyword is detected, the master node will trigger a function and call the corresponding services; If a open question is detected, the master node will search the answers. Moreover, the master node also handle `detected_fall` by triggering an active query. The PC holds all the services and handles the articulation of Pepper. The following sections explain how Pepper speaks, the Q&A, and all the services.

1) Speech Generation: To make Pepper speak, all the audio responses are published to the`pepper_talker` topic, including active queries, answers to the questions, and feedback on the services. Then we utilise the `pepper_listener` node and the ALTextToSpeech API from PC to allow Pepper articulate.

The listener node subscribes to the `pepper_talker` topic and send any word string received to the ALTextToSpeech module, which allows the robot to speak. The ALTextToSpeech sends commands to a text-to-speech engine,

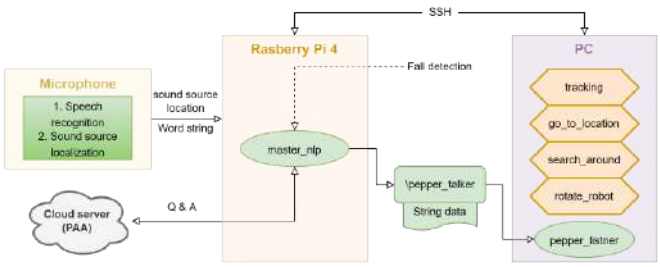

Fig. 9. Design architecture of the interactive state

and authorizes also voice customization. The result of the synthesis is sent to the robot's loudspeakers.

2) Questions and Answers: If the words string received by the microphone is a question, the master node will trigger the Q&A module. We treat the queries as open questions and search answers using google search through People Also Ask (PAA) APIs. Even if the search result does not have a direct answer for the question, the APIs can also return a list of questions which are either most related or most popular. The Q&A system can answer questions of any type with high variance, including What, When, How, Where, etc. A answering example for health related question is shown in figure 10

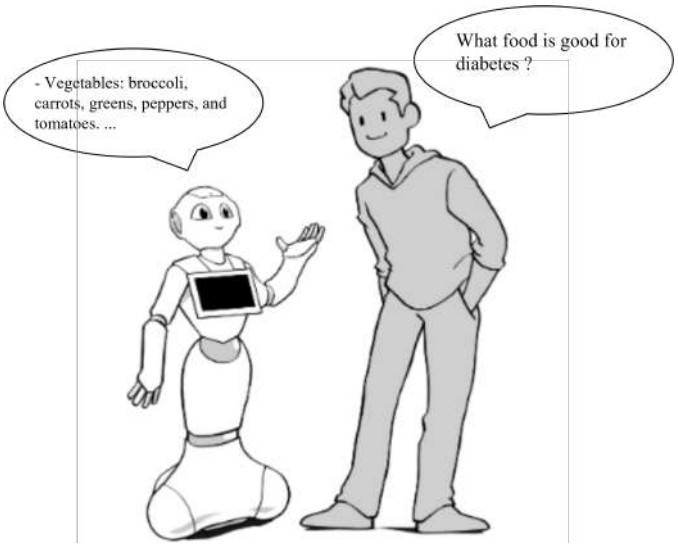

Fig. 10. Questions and Answers

E. Services

Functions are achieved by calling a series of services. This section revisits all the tasks and explains what services are used.

1) Follow Me: This function calls the `tracking` service. In this service, to make Pepper follows people, we utilise the ALTracker module. It allows the robot to track different targets (red ball, face, landmark, etc.) using different means (head only, whole body, move, etc.). The main goal of this module is to establish a bridge between target detection and motion in order to make the robot keep in view the target in the middle of the camera.

2) Come Here: The first service involved in this function is `rotate_robot`, which takes the sound source angle returned by the microphone and make Pepper turn to that direction using the ALNavigation API. ALNavigation API allows the user to perform safe displacements when using the robot.

The second service is `tracking`, which detects the human and approaches the target. Come Here lets Pepper stop automatically in front of the user at a safe distance.

3) Find Object: The first service involved in this function is `find_object_location`. Essentially this service takes any class label as a request and returns two pieces of information: The `isInFrame` variable tracks whether the required object is in the current frame. A position variable maintains the location of Pepper (w.r.t the world frame) when that object is detected last time. If the object has not been detected before, the position is set to a default value (0).

After the master NLP node extracts the object to be found from the query, it will send a request to the `find_object_location` service with the object class name, checking whether the object has been detected. If not, Pepper will inform the user and end the query. If the object has been detected, Pepper will navigate to the recorded position using the `go_to_location` service.

Next, Pepper will call the `find_object_location` service again to check whether the object is in the current frame. If not, the `search_around` service will be called subsequently. It turns Pepper 360 degrees to search for the target, using the `rotate_robot` service, until the object is in the frame.

Finally, Pepper will inform the user and terminate the query after the required object appears in the current frame.

VII. Experiment and Result

A. Experimental Setup

Four experiments were carried out in Imperial College London's room EEE505 to test all of the mentioned hypotheses. All survey participants were chosen from Imperial's EEE department. This is because, during a pandemic, finding elderly people living in single-occupancy rooms in their care homes to test our robot is difficult.

Before starting the experiments, CareBot has navigated in the given environment for the initialisation of some functionalities, i.e., memorising the location of the objects. Participants were provided advance instructions on how to operate the robot's various functionalities in all studies. Following that, the participants could interact with CareBot based on the different experimental criteria. Some quantitative terms (i.e., success rate) were used to evaluate CareBot's performance.

B. Hypothesis Validation

1) ***H1 Experiments****:* For H1, the participant has interacted with CareBot to chat for a while. At the beginning, the user

will ask the CareBot to 'Come Here' and the CareBot will come and stop in front of the user at the distance of around 0.5 meters. Then the user could begin chatting with the CareBot. The chat includes but not limited to asking for the weather, and the date today. The CareBot would search online and reply the user.

Results During 10 trials on the first experiment, The CareBot could come to the user in 8 times. It has successfully recognised and replied for 85% questions. The failures may because of the noisy environment.

*2) **H2 Experiments**:* To evaluate H2, participants ask the robot to locate 5 items: bottle, umbrella, remote, mouse, and phone. Each item is placed in a distinct area in the room that CareBot is able to see. As the function of find object is continuously running in the background mode, the CareBot would 'remember' the location of the object, when it appears in pepper's sight. Thus, the experiment began with placing an object in the area that CareBot could detect. In this stage, the participant would ask the CareBot to follow him/her and change its location in the room. Participants will next ask the CareBot to search for the objects. After the CareBot navigate to the position of the object, it will turn around and find the object so that the user is able to find it.

Results During our experiments, many fatal happened because there were obstacles around the CareBot and stopped it to turn or move. Despite that, over 50 trials during our experiment, it successfully directed to the location and found 23 objects.

*3) **H3 Experiments**:* In the last experiment, participants will simulate falling events in the sight of CareBot 50 times at different distances. The number of successfully detected falling events will be used to evaluate if the hypothesis H3 is held. The visualization of fall detection is shown in figure 11.

Fig. 11. Visulaization of Fall vs Normal

Results For the real environment test, falls can be detected successfully 49 out of 50 trials, achieving 98% test accuracy. Furthermore, the fall detection system can operate in real-time with an average inference time of $0.095s$, confirming our premise of lowering the risk of older people following a fall, as the robot can care the elderly people in time. To further improve the user experience, an extended function to assist the elderly as they fall can be incorporated into our system in the future. For example, the robot could send a message to or call his/her children straightaway to provide the elderly with timely medical assistance.

VIII. Challenges

This part mainly discusses the challenges we have faced and cannot be resolved during the experiments in the real scene. We tried to utilize the ROS 2D navigation stack as the moving module in our project. However, many failures occur, and the following paragraphs show the detail.

A. Navigation & SLAM

Having accurate navigation is helpful for CareBot to serve its client, and many functions such as object funding highly rely on it. To balance the computational power required for the computer and efficiency, ROS 2D mapping is selected. ROS `gmapping` and `move_base` nodes are wildly used within the SLAM implementation, but several problems happen when using them with Pepper's Odom and laser scan system. A further problem will be broken down into the following session: Mapping, AMCL, Navigation and another approach.

1) Mapping: Data from the Odom (robot relative position) and the laser scanner are combined to create a 2D map from scratch with Pepper. However, IMUs, encoders and motors from Pepper has drifting problem, so that Pepper cannot locate itself as shown in figure 12. To solve this issue, a clean map is reproduced with image editing software.

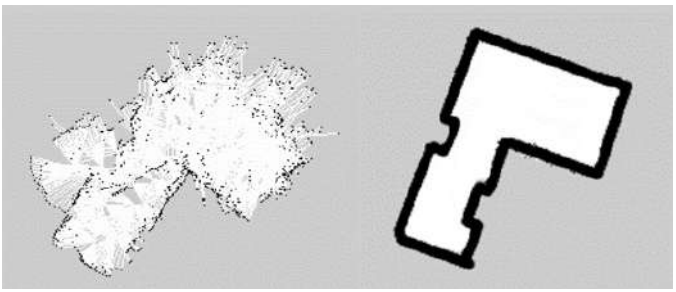

Fig. 12. Shift Issue (left), Clean Map (right)

2) AMCL: A clear map is essential for the robot to localise itself. With a confusing map (left of figure 13), the localisation program cannot generate reasonable localization results. This is shown by the spread of the red arrow across the map. After using the manually created map, the localization program can localize the robot correctly with confidence as shown with the spread of the red arrow.

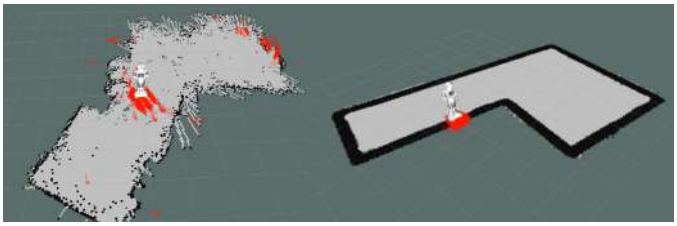

Fig. 13. Localization with Confusing Map (left), Localization with Clear Map (right)

3) Navigation / Navigation with r-TAB: When the robot navigates across the map, other than receiving the target, accuracy feedback from Odom and the laser scanner is essential to let the navigation program calculate the navigation path. By navigating using ROS `move_base` with a 2D map after localizing itself, the Pepper can navigation avoid obstacle during the navigation. However, with the drifting issue, Pepper sometimes would confuse its location and non-exist obstructed object (left of Figure 14), which will affect the performance of the navigation process. On the other hand, navigation with r-TAB can solve this issue introduced by Odom by combining a visual & depth camera and a laser scanner to replace localization with Odom. However, since r-TAB involves visual & depth image processing, it requires more computational capability to achieve the navigation process, or the 3D map cannot be created successfully as shown in the right of Figure 14.

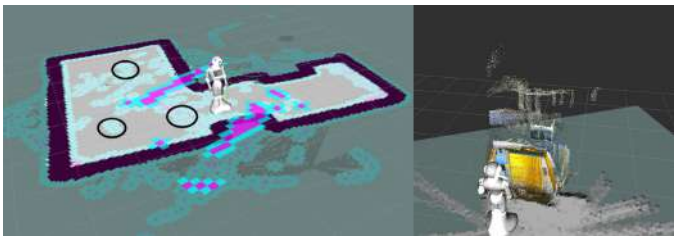

Fig. 14. Navigation in 2D (left), 3D Map Create with R-TAB (right)

4) Patrol: Patrol is a service originally designed for searching the locations of target object if it does not exist in the current database when CareBot is required to find it. However, Patrol was not able to implement due to the fact that Pepper robot cannot navigate in the map.

A theoretical simulation was implemented using AMCL and `move_base` nodes in both STDR simulator and Rviz (a short video can be referred to GitHub link). The virtual robot in simulator can accurately navigate to pre-defined coordinates within the pre-built Rviz map using path planning, as shown in figure 15. The coordinates of target points can be acquired by manually setting the 2D navigation goal in the Rviz map. The `move_base` node provides path planner which constantly updates the path to avoid obstacles, ensuring virtual robot safely navigate to the target goal without collision.

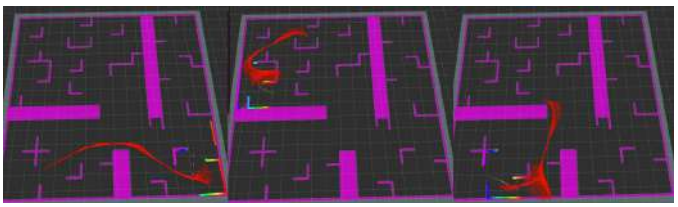

Fig. 15. Patrol Simmulation Running in Rviz

An alternative approach for patrol based on ALMotion APIs was implemented, with which the movement to fixed point can be achieved by defining the relative coordinates and rotating angle for the input arguments. However, the existing APIs is not an ideal solution for patrol as it would navigate to the defined target point regardless of obstacles in reality which would potentially lead to collision. Therefore, with this limitation, the patrol service can only be run on the empty space. Further improvement for patrol would be investigated in future study.

IX. Conclusion and Future work

In this report, CareBot is proposed as an intelligent assistant and home companion, providing a series of auxiliary functions and human-robot interaction (HRI) for the elderly, particularly in home-care scenarios. We successfully implement functions including calling the CareBot to come closer and follow the user, asking the CareBot to find a predefined object in the environment, and monitoring the user's pose for fall detection. Two states and five functions are explained in detail and evaluated in the previous sections. We also illustrate the main challenge - navigation and shed some light on how we address the issue and the remaining problems. In sum, we achieve the goal set in the design report with great effort, and most of the hypotheses are verified.

We still see a great potential to extend the project. Here we list some future works that might be done:

- Q&A has been implemented for everyday conversation. However, a reminder function can be implemented on top: the CareBot would remind the user to drink water or take medicines at the appointed time.
- The object detection can be improved by retaining a new Tiny YOLO v4 model on a larger dataset that includes objects the elderly are more likely to encounter in daily life.
- The database can be extended to store more information.
- Other actions recognition like sedentary and lying analysis can be integrated for human-robot interactions rather than only fall detection.
- Pepper is capable of moving arms; more humanoid actions can be added, such as pointing to the found object.

All the code and a demo video can be found in the Github repository :

Fig. 16. Github link: https://github.com/Yebulabula/HCR

REFERENCES

[1] Age UK, "Three-quarters of over 65s admit they're worried about rising cost of living," 2022, [Online]. Available: https://www.ageuk.org.uk/latest-press/articles/2022/three-quarters-of-over-65s-admit-theyre-worried-about-rising-cost-of-living/.

[2] D. Feil-Seifer and M. J. Matarić, "Socially assistive robotics," *IEEE Robotics Automation Magazine*, vol. 18, no. 1, pp. 24–31, 2011.

[3] J. Broekens, M. Heerink, and H. Rosendal, "Assistive social robots in elderly care: A review," *Gerontechnology*, vol. 8, pp. 94–103, 04 2009.

[4] J. Jones, "Robots at the tipping point: the road to irobot roomba," *IEEE Robotics Automation Magazine*, vol. 13, no. 1, pp. 76–78, 2006.

[5] M. Pollack, L. Brown, D. Colbry, C. Orosz, B. Peintner, S. Ramakrishnan, S. Engberg, J. Matthews, J. Dunbar-Jacob, C. Mccarthy, M. Montemerlo, J. Pineau, and N. Roy, "Pearl: A mobile robotic assistant for the elderly," 06 2002.

[6] K. Wada, T. Shibata, T. Saito, and K. Tanie, "Effects of robot-assisted activity for elderly people and nurses at a day service center," *Proceedings of the IEEE*, vol. 92, no. 11, pp. 1780–1788, 2004.

[7] H. Hodson, "The first family robot," 2014.

[8] T. Klamer and S. B. Allouch, "Acceptance and use of a social robot by elderly users in a domestic environment," in *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*. IEEE, 2010, pp. 1–8.

[9] G. Milliez, "Buddy: A companion robot for the whole family," in *Companion of the 2018 ACM/IEEE international conference on human-robot interaction*, 2018, pp. 40–40.

[10] M. Fujita, "On activating human communications with pet-type robot aibo," *Proceedings of the IEEE*, vol. 92, no. 11, pp. 1804–1813, 2004.

[11] C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, and S. Šabanović, *Human-robot interaction: An introduction*. Cambridge University Press, 2020.

[12] J. Broekens, M. Heerink, H. Rosendal *et al.*, "Assistive social robots in elderly care: a review," *Gerontechnology*, vol. 8, no. 2, pp. 94–103, 2009.

[13] B. Graf, M. Hans, and R. D. Schraft, "Care-o-bot ii—development of a next generation robotic home assistant," *Autonomous robots*, vol. 16, no. 2, pp. 193–205, 2004.

[14] A. K. Pandey and R. Gelin, "A mass-produced sociable humanoid robot: Pepper: The first machine of its kind," *IEEE Robotics & Automation Magazine*, vol. 25, no. 3, pp. 40–48, 2018.

[15] Luxonis, "Oak-d documentation," 2021. [Online]. Available: https://docs.luxonis.com/projects/hardware/en/latest/pages/BW1098OAK.html

[16] Intel, "Openvino documentation," 2021. [Online]. Available: https://docs.openvino.ai/latest/index.html

[17] I. Khokhlov, E. Davydenko, I. Osokin, I. Ryakin, A. Babaev, V. Litvinenko, and R. Gorbachev, "Tiny-YOLO object detection supplemented with geometrical data," *arXiv e-prints*, p. arXiv:2008.02170, Aug. 2020.

[18] J. Han, E. Haihong, G. Le, and J. Du, "Survey on nosql database," in *2011 6th international conference on pervasive computing and applications*. IEEE, 2011, pp. 363–366.

[19] R. Bajpai and D. Joshi, "Movenet: A deep neural network for joint profile prediction across variable walking speeds and slopes," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2021.